HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

DIPLOMARBEIT

Herr
Stefan Sailer

Auswerten von CPU-gestützten Algorithmen zur Störsignalunterdrückung im GNSS-Bereich und FPGA-Implementierung eines SDRs

Mittweida, 2015

Fakultät Elektro- und Informationstechnik

DIPLOMARBEIT

Auswerten von CPU-gestützten Algorithmen zur Störsignalunterdrückung im GNSS-Bereich und FPGA-Implementierung eines SDRs

Autor:
Stefan Sailer

Studiengang:
Informationstechnik

Seminargruppe:
KI10w1-F

Erstprüfer:
Prof. Dr.-Ing. Alexander Lampe

Zweitprüfer:
M.Sc. Thomas Kraus

Einreichung:
Mittweida, 12.05.2015

Verteidigung/Bewertung:
München, 2015

Bibliografische Angaben:

Sailer, Stefan:
Auswerten von CPU-gestützten Algorithmen zur Störsignalunterdrückung im GNSS-Bereich und FPGA-Implementierung eines SDRs. - 2015 -, 14 Seiten Verzeichnisse, 75 Seiten Inhalt, 8 Seiten Anlagen, 54 Abbildungen
Mittweida, Hochschule Mittweida (FH), University of Applied Sciences, Fakultät Elektro- und Informationstechnik, Diplomarbeit 2015

Referat:

Im Rahmen dieser Diplomarbeit werden CPU-gestützte Algorithmen zur Störsignalunterdrückung im GNSS-Bereich auf ihre Wirksamkeit hinsichtlich verschiedener Störsignale untersucht. Vor der Untersuchung müssen die störbehafteten GNSS-Signale im Labor unter Einhaltung vorliegender Bedingungen aufgezeichnet werden. Abschließend wird der effektivste Algorithmus auf der FPGA-Technologie umgesetzt. Dazu dient ein SDR-System mit integrierten FPGA als GNSS-Störsignalunterdrückungseinheit. Das SDR-System fungiert letztendlich als GNSS-Front-End mit Fähigkeit zur Störsignalunterdrückung für einen GNSS-Softwareempfänger oder als Störsignalunterdrückungsmodul für eigenständige GNSS-Empfänger.

Danksagung

Diese Diplomarbeit wurde am Institut für Raumfahrttechnik und Weltraumnutzung an der Universität der Bundeswehr München und an der Hochschule Mittweida für den Studiengang Informationstechnik (FH) angefertigt.

An dieser Stelle möchte ich mich vielmals bei allen Personen bedanken, die mir bei der Realisierung dieser Diplomarbeit zur Seite standen.

An erster Stelle möchte ich mich bei meinem Hochschulbetreuer Herrn Prof. Dr.-Ing. Alexander Lampe bedanken. Er gab mir die Möglichkeit, diese Diplomarbeit unter seiner Leitung anzufertigen.

Besonderer Dank gilt an dieser Stelle meinem Betreuer M.Sc. Thomas Kraus, für die interessante Aufgabenstellung aus einem aktuellen Forschungsgebiet der Satellitennavigation. Durch seine große Unterstützung mit viel Engagement und Zeitaufwand, Anregungen und guten Ideen wäre diese Arbeit nicht möglich gewesen.

Ebenfalls möchte ich mich bei Herrn Prof. Dr.-Ing. Bernd Eissfeller bedanken. Er hat es mir ermöglicht, diese Arbeit an seinem Lehrstuhl für Satellitennavigation zu erstellen und mich stets gefördert.

Großer Dank gelten meiner Kollegin Dipl.-Soz. Kristina Kudlich und meiner Schwester Katharina für das Korrekturlesen.

Ein großer Dank gilt meinen Eltern für ihre Geduld und immerwährende Unterstützung.

Stefan Sailer

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADC	Analog-to-Digital-Converter
AGC	Automatic Gain Control
ARNS	Aeronautical Radio Navigation Service
ASIC	Application-specific integrated circuit
AWGN	Additives White Gaussian Noise
BH	Blackmann Harris
BW	Band Width
C/A	Coarse/Acquisition
C/N_0	Carrier-to-noise ratio
CDMA	Code Division Multiple Access
CLB	Configurable Logic Block
CPU	Central Processing Unit
CW	Continuous Wave
DAC	Digital-to-Analog-Converter
DFT	Discrete Fourier-Transform
DLL	Dynamic Link Library
DMA	Direct Memory Access
DME	Distance Measuring Equipment
DRAM	Dynamic Random Access Memory
DVB-T	Digital Video Broadcasting - Terrestrial
EMVG	Gesetzt über die elektromagnetische Verträglichkeit von Betriebsmitteln
ESA	European Space Agency
FDMA	Frequency Division Multiple Access
FFT	Fast Fourier-Transform
FIFO	First In First Out
FM	Frequenzmodulation
FPGA	Field Programmable Gate Array
FWHM	Full width at half maximum
FXP	Fixed-Point
GLONASS	GLObal NAVigation Satellite System
GmbH	Gesellschaft mit beschränkter Haftung
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HA	Hann

HAM	Hamming
HDL	Hardware Description Language
HF	Hochfrequenz
I/Q	In-Phase-&-Quadrature-Verfahren
IF	Intermediate Frequency
IFFT	Inverse Fast Fourier Transform
IG	Interferenzen-Generator
IOV	In Orbit Validation
Kfz	Kraftfahrzeug
KLT	Karhunen-Loève-Transformation
LabVIEW	Laboratory Virtual Instrumentation Engineering
LTE	Long Term Evolution
LUT	Lookup Table
NCO	Numerically Controlled Oscillator
NI	National Instruments
NMEA	National Marine Electronics Association
OCXO	Oven Controlled Crystal Oscillator
OV	Overlapping
P/Y	Precision/Encrypted
PPDs	Personal Privacy Devices
PRN	Pseudo Random Noise
PRS	Public Regulated Service
PXI	PCI Extensions for Instrumentation
PXIe	PCI Extensions for Instrumentation express
QMF	Quadrature Mirror Filter
Radar	Radio Detection and Ranging
RAM	Random Access Memory
RE	Rectangular
RINEX	Receiver Independent Exchange Format
SA	Selective Availability
SAR	Search And Rescue
SDR	Software Defined Radio
SRAM	Static Random Access Memory
SSH	Secure Shell
STFT	Short Time Fourier Transformation
TCXO	Temperature Compensated Crystal Oscillator
TDK	TERMINATE Development Kit

TKG	Telekommunikationsgesetz
USA	United States of America
USRP-RIO	Universal Software Radio Peripheral – Reconfigurable Input/Output
VI	Virtuelle Instrumente
VSA	Vektorsignalanalysator
VSG	Vektorsignalgenerator
WT	Wavelet-Transformation
ZF	Zwischenfrequenz

1 Übersicht

Das erste Kapitel stellt eine Übersicht über den Inhalt dieser Diplomarbeit dar. Hierbei wird die Motivation des bearbeiteten Themas erläutert sowie die Zielsetzung festgelegt. Abschließend erfolgt eine Kapitelübersicht.

1.1 Motivation

Die Benutzung eines globalen Satellitennavigationssystems (engl.: Global Navigation Satellite System; Abk.: GNSS) gewinnt immer mehr an Bedeutung. Viele Bereiche sind in der heutigen Zeit für die Navigation auf ein solches System angewiesen, so dass es immer wichtiger wird, eine uneingeschränkte und ausreichende Verfügbarkeit sicherzustellen. Dies kann jedoch durch verschiedene Störungen beeinträchtigt werden, die den Empfang der Satellitensignale verschlechtern oder gar unmöglich machen, was eine ungenaue oder gar keine Positionsbestimmung zur Folge hat. Diese Störsignale können unabsichtlich oder absichtlich herbeigeführt werden. Eine unabsichtliche Störung geht von Systemen aus, welche im selben Frequenzband wie GNSS arbeiten, aber auch von Fehlfunktionen anderer Systeme außerhalb des GNSS-Frequenzbandes. Absichtliche Störungen werden durch GNSS-Störsender aus verschiedenen Gründen bewusst herbeigeführt. Das Verschleiern des eigenen Aufenthaltsortes, außer Kraft setzen von GNSS-basierten Kfz-Diebstahlfunktionen und mutwillige Betrugsversuche gegen GNSS-basierte Mautsysteme können zum Beispiel der Grund dafür sein. Bei diversen Forschungsvorhaben werden dazu Gegenmaßnahmen getroffen, welche zum Erkennen und Unterdrücken von Störsignalen dienen und trotz auftretender Störungen eine genaue Navigation möglich machen. Eine Möglichkeit, einer Signalverschlechterung durch Störungen entgegenzuwirken, besteht in der Anwendung von Algorithmen zur Störsignalunterdrückung. Da diese Algorithmen auf einem Rechner für den Prozessor (engl.: Central Processing Unit; Abk.: CPU) eine sehr rechenintensive Aufgabe darstellen, wird die Realisierung solcher auf einem programmierbaren integrierten Schaltkreis (engl.: Field Programmable Gate Array; Abk.: FPGA) angestrebt. Dieser bietet durch seine parallele Verarbeitung eine schnellere und effizientere Möglichkeit für die Implementierung von Algorithmen zur Störsignalunterdrückung.

1.2 Zielsetzung

Die Zielsetzung lässt sich in zwei Gruppen aufteilen. Der erste Teil befasst sich mit dem Auswerten von CPU-gestützten Algorithmen zur Störsignalunterdrückung im GNSS-Bereich. Um die Wirkung der vorliegenden Algorithmen auf die im GNSS-Signal enthaltene Störung auszuwerten, muss dieses als digitale Signalfolge vorliegen und zuvor aufgezeichnet werden. Sind diese vorhanden, werden die verschiedenen Algorithmen mit auswählbaren Parametern auf das jeweilige GNSS-Signal mit Störkomponente angewendet. Mittels der auswählbaren Parameter gilt es, die effektivsten Einstellungen zu

ermitteln. Anhand der Auswertungen erfolgt eine Übersicht des am besten geeigneten Algorithmus für die jeweils vorliegende Art der Störung. Im zweiten Teil wird der effektivste Algorithmus für alle behandelten Störsignale mit LabVIEW als Programmiersprache auf einem FPGA eines Software Defined Radio (SDR) implementiert, welches als Front-End für einen GNSS-Softwareempfänger oder GNSS-Repeater dienen kann. Durch diese Umsetzung wird abschließend eine GNSS-Störsignalunterdrückungseinheit, mit der die Störkomponenten im GNSS-Signal in Echtzeit unterdrückbar sind realisiert.

1.3 Kapitelübersicht

Das **erste Kapitel** stellt die Übersicht der Diplomarbeit dar und beinhaltet Motivation, Zielsetzung und Kapitelübersicht. Der Abschnitt „Motivation“ erläutert die Hintergründe für die Auswahl dieses Themas. Die Zielsetzung stellt eine grobe Übersicht über die angestrebten Ziele dar.

Im **zweiten Kapitel** werden die benötigten Grundlagen dargestellt. Als erstes erfolgt ein Überblick über die meistverwendeten GNSS-Systeme und eine kurze technische Erläuterung der Funktionsweise von solchen Systemen. Danach wird die Arbeitsweise von GNSS-Empfängern erläutert und der aktuelle Stand zu den bereits verfügbaren Unterdrückungsmethoden dargestellt. Abschließend erfolgt ein Überblick über vorkommende Störquellen.

Im **dritten Kapitel** erfolgt die Präzisierung der Aufgabenstellung, welche sich in drei Hauptfelder gliedern lässt. Am Anfang müssen die störbehafteten GNSS-Signale bereitgestellt werden. Diese werden aufgenommen und als Samples abgespeichert, damit diese zur weiteren Verarbeitung zur Verfügung stehen. Sind alle Szenarien aufgezeichnet, kann mit der Analyse zur Erkennung und Unterdrückung von Störsignalen im GNSS-Bereich begonnen werden. Abschließend wird die Umsetzung der GNSS-Störsignalunterdrückungseinheit präzisiert.

Das **vierte Kapitel** beschäftigt sich mit dem Aufzeichnen von störbehafteten GNSS-Signalen, welche für die Auswertung bereitstehen müssen. Dazu wird die verwendete Hardware und Software vorgestellt und die Vorgehensweise erläutert.

Das **fünfte Kapitel** widmet sich dem Unterdrücken von Störquellen. Die Hauptaufgabe liegt darin, für die verschiedensten möglichen Störquellen die besten Algorithmen mit Parametern zu bestimmen, auszuwerten und darzustellen. Außerdem wird sich auf die Ermittlung der Echtzeitfähigkeit der Algorithmen bezogen.

Das **sechste Kapitel** beschreibt den Vorgang der Realisierung der GNSS-Störsignalunterdrückungseinheit auf einem SDR mit FPGA. Neben den Anforderungen wird die Umsetzung erläutert. Abschließend erfolgen Tests über die Funktionsweise.

Im **siebten Kapitel** folgt das Schlusswort und gibt einen Ausblick über zukünftige Themen.

2 Grundlagen und Stand der Technik

2.1 Einleitung

In diesem Kapitel werden die notwendigen theoretischen Grundlagen erläutert und der Stand der Technik dargestellt. Als erstes erfolgt ein Überblick über die meistverwendeten GNSS-Systeme und eine kurze technische Erläuterung der Funktionsweise. Danach wird die Arbeitsweise eines GNSS-Empfängers erläutert und der aktuelle Stand zu den bereits verfügbaren Unterdrückungsmethoden dargelegt. Abschließend erfolgt ein Überblick über vorkommende Störquellen.

2.2 Überblick der GNSS-Systeme

Ein GNSS ist ein System zur Positionsbestimmung mittels Satelliten, welche sich in der Erdumlaufbahn befinden. Das wohl bekannteste GNSS ist das Globale Positionsbestimmungssystem (engl.: Global Positioning System; Abk.: GPS), das von den USA seit den 1970er-Jahren für militärische Zwecke entwickelt und verwendet wird. Nach Abschalten der künstlichen Signalverschlechterung im Jahre 2000 wird GPS auch in zivilen Bereichen (z.B. Luftfahrt, Seefahrt, etc.) als Mittel zur Navigation genutzt. Durch die zunehmende Bedeutung dieses Systems und die daraus resultierende Abhängigkeit betreiben seit einigen Jahren auch andere Nationen ein eigenes GNSS oder befinden sich im Aufbau eines solchen, um die Unabhängigkeit von den USA zu erreichen. Die europäische Union plant und entwickelt das System „Galileo“, welches sich derzeit noch in der Aufbauphase befindet. Galileo ist für eine rein zivile Nutzung vorgesehen und soll mehrere Dienste für die Nutzung bereitstellen. Neben den Europäern betreibt Russland GLONASS, welches bereits in Betrieb ist. China befindet sich mit BeiDou ebenfalls wie Europa im Aufbau seines eigenen GNSS.

Tabelle 1: GNSS-Systeme (Quelle: [1])

System	Betreiber	Erste Generation	Zweite Generation
NAVSTAR-GPS	USA	seit 1995	ab 2012
GLONASS	Russland	seit 1993	ab 2012
Galileo	Europa	geplant ab 2017	
QZSS	Japan		
Beidou	China		

GNSS-Satelliten senden ein kontinuierliches Navigationssignal aus, welches von Empfängern für die Positionsbestimmung empfangen wird. Mit der im Signal enthaltenen Nachricht erfolgt eine Positionsbestimmung im Empfänger. Für den Betrieb eines GNSS sind mehrere Einheiten notwendig, welche sich bei GPS in drei Hauptsegmente gliedern lassen: Das Weltraum-, Kontroll- und Nutzersegment. Das Weltraumsegment besteht aus den in der Erdumlaufbahn befindlichen Satelliten. Im Kontrollsegment werden die Satelliten

überwacht, gesteuert und mit den notwendigen Informationen beliefert. Das Nutzersegment umfasst die Anwender in Form von Empfängern [2]. Abbildung 1 stellt die verwendeten Frequenzen für aktuelle GNSS dar. Die blau hervorgehobenen Felder deuten auf aktuell funktionsfähige Satellitensignale hin.

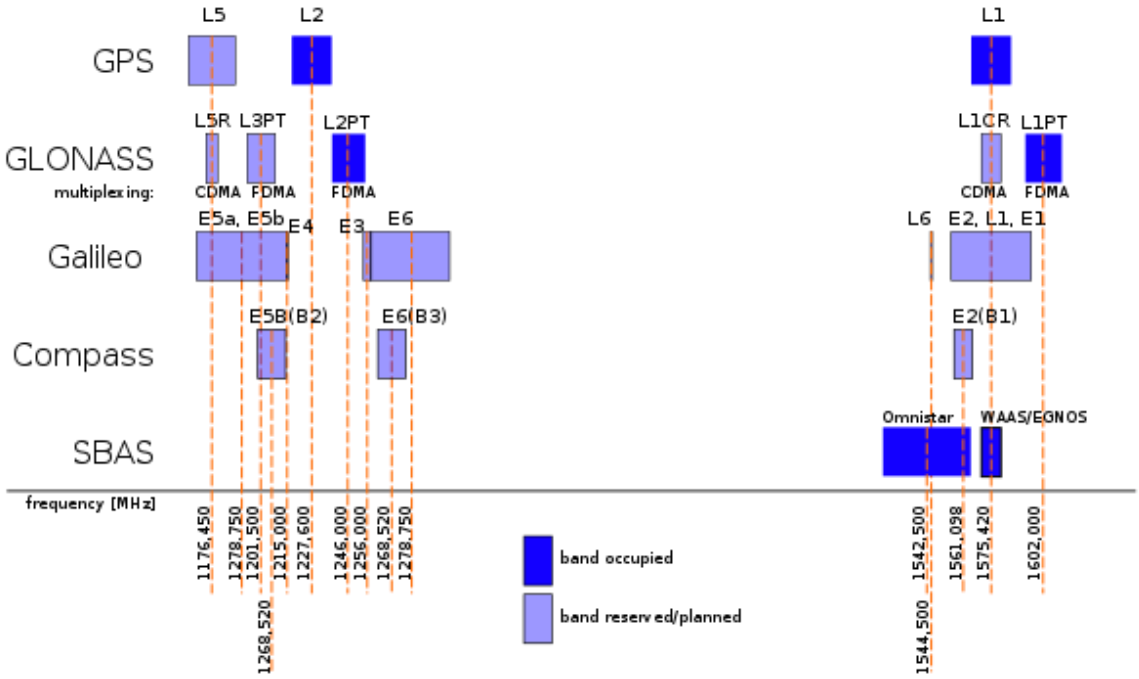


Abbildung 1: Frequenzen der verschiedenen GNSS (Quelle: [1])

2.2.1 GPS

Das NAVSTAR GPS (Navigational Satellite Timing and Ranging – Global Positioning System), von den USA in den 1970er-Jahren für militärische Zwecke entwickeltes GNSS-System, besteht derzeit aus 31 Satelliten [3], welche die Erdumlaufbahn in einer Höhe von 20.200 km umkreisen. Seit 1995 ist dieses System offiziell in Betrieb. Für eine ausreichende Navigation auf der Erde müssen immer vier der Satelliten für den Empfänger sichtbar sein, weshalb mindestens 24 Satelliten gültige Signale aussenden müssen. Durch Abschalten der künstlichen Signalverschlechterung (engl.: Selective Availability; Abk.: SA) ist seit dem Jahr 2000 GPS auch für zivile Nutzer verfügbar [2]. Seit 2004 besteht die Übereinkunft von den USA mit der EU, GPS und Galileo für zivile Zwecke mit einer gemeinsamen Signalmodulation zu verwenden. Somit kann eine bessere Zeitbestimmung, Navigation und Positionsbestimmung erzielt werden [4].

2.2.2 Galileo

Galileo ist das von der Europäischen Union entwickelte GNSS-System, welches sich derzeit noch im Aufbau befindet. Bis zu der Fertigstellung sollen sich 30 Satelliten in der Erdumlaufbahn von etwa 23.260 km Höhe befinden. Momentan sind acht Satelliten in der Erdumlaufbahn, wobei drei funktionsfähig sind [Stand: Januar 2015]. Die zwei ersten Testsatelliten GIOVE-A und B wurden für erste Versuche und Testzwecke ins Weltall geschickt. Diese haben ihre Lebensdauer erreicht und sind nicht mehr in Betrieb. Nach

erfolgreicher Erprobung wurden im Jahre 2011 die ersten vier funktionsfähigen IOV (in-orbit validation) Satelliten in den Weltraum befördert, womit die erste Phase 2012 abgeschlossen wurde. Die letzten beiden Satelliten wurden aufgrund eines technischen Fehlers in der Trägerrakete im falschen Orbit ausgesetzt. Die erste operationelle Phase ist bis 2016 geplant. Neben dem Open Service mit kostenlosem Empfang der Navigationssignale ist für das Galileo-System noch ein kostenpflichtiger kommerzieller Dienst, mit welchem sich eine genauere Positionsbestimmung erreichen lässt, geplant. Zusätzlich sind noch die Dienste PRS und SAR geplant [2].

2.2.3 GLONASS

GLONASS (GLObal Navigation Satellite Systeme) ist das GNSS der Russischen Föderation, welches im Vollausbau aus 21 Satelliten besteht. Die Entwicklung dieses Systems begann im Jahre 1972 und ist im Jahre 1993 offiziell in Betrieb gegangen. Nach dem Aufbau war GLONASS bis zum Jahre 2011 nicht vollständig funktionsfähig. GLONASS nutzt im Vergleich zu GPS und Galileo andere Frequenzen für die Übertragung der Navigationsdaten sowie ein anderes Modulationsverfahren. Während GPS und Galileo mit dem Codemultiplexverfahren (engl.: Code Division Multiple Access; Abk.: CDMA) arbeiten, verwendet GLONASS dafür das Frequenzmultiplexverfahren (engl.: frequency-division multiple access; Abk.: FDMA). Jedoch benutzt GLONASS das Codespreizverfahren mit einer PRN-Sequenz zum Erreichen eines Korrelationsgewinns [5].

2.2.4 Funktionsweise von GNSS am Beispiel von GPS

Die Funktionsweise von einem GNSS wird anhand von GPS kurz erläutert und dargestellt. Ein GPS-Signal besteht aus drei Komponenten. Diese sind der Träger, die Navigationsdaten und der Code, welcher auf dem Satelliten erzeugt wird. Abbildung 3 zeigt den schematischen Ablauf dieses Vorgangs. Das GPS-Signal kann mit zwei verschiedenen HF-Signalen im L-Band, welches seinen Frequenzbereich von 1 GHz bis 2 GHz hat, übertragen werden. Diese Frequenzen der Trägerwelle heißen L1 und L2 und sind abgeleitet von einer gemeinsamen Grundfrequenz f_0 = 10,23 MHz:

$$f_{L1} = 154 * f_0 = 1575{,}42\ MHz$$

$$f_{L2} = 120 * f_0 = 1226{,}60\ MHz$$

Neben den bereits verwendeten Frequenzen ist aktuell noch die Erweiterung um das L5 Band im Aufbau, um die Robustheit von GPS weiter zu verbessern. Die Navigationsdaten enthalten Informationen über die Satellitenbahnen, welche über die Bodenstation aktuell gehalten werden. Die Bitrate beträgt 50 Bit/s. Jeder Satellit hat zwei einzigartige Codes: Einen Akquirierungscode (engl.: Coarse/Acquisition-Code; Abk.: C/A-Code) mit einer Länge von 1023 Bit und einen verschlüsselten militärischen P/Y-Code (engl.: Precision/encrypted), welcher eine genauere Positionsbestimmung ermöglicht. Der C/A-Code wird nur über das L1-Band und der P/Y-Code über beide Frequenzbänder L1 und L2 übertragen. Mittels Phasenmodulation erfolgt die Modulation des Codes auf den Träger.

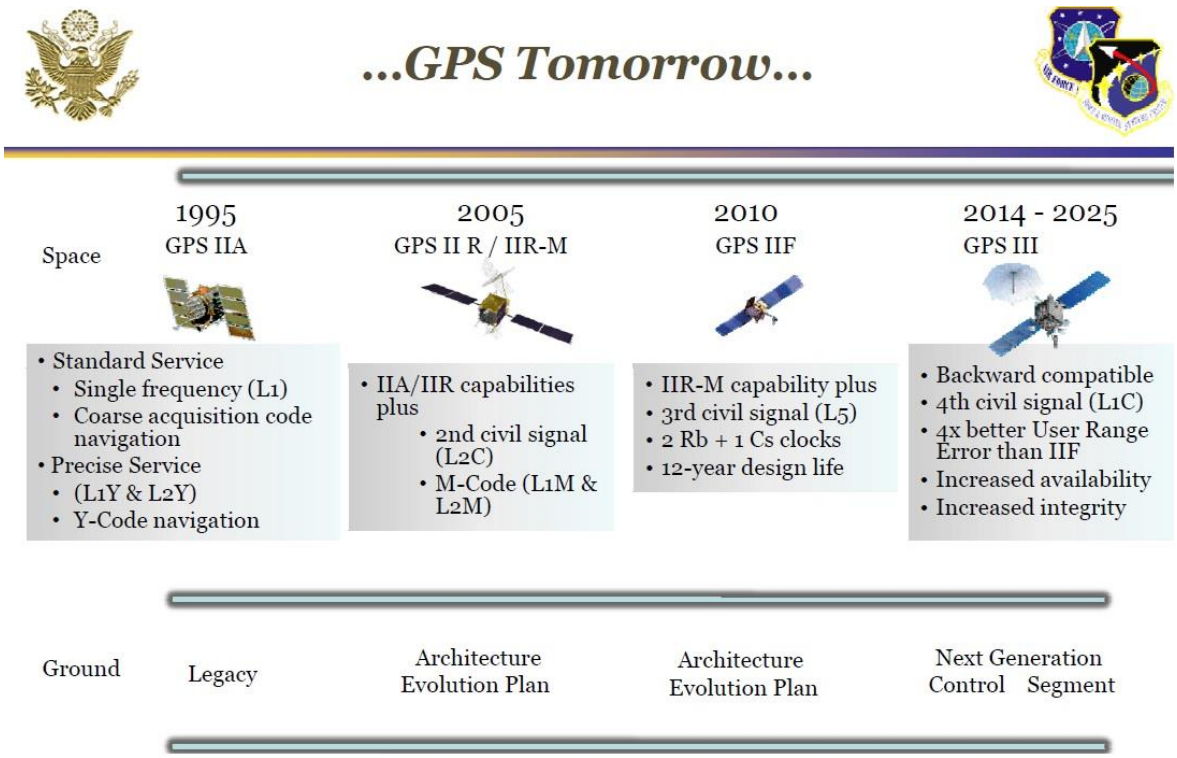


Abbildung 2: Entwicklungsphase von GPS (Quelle: [6])

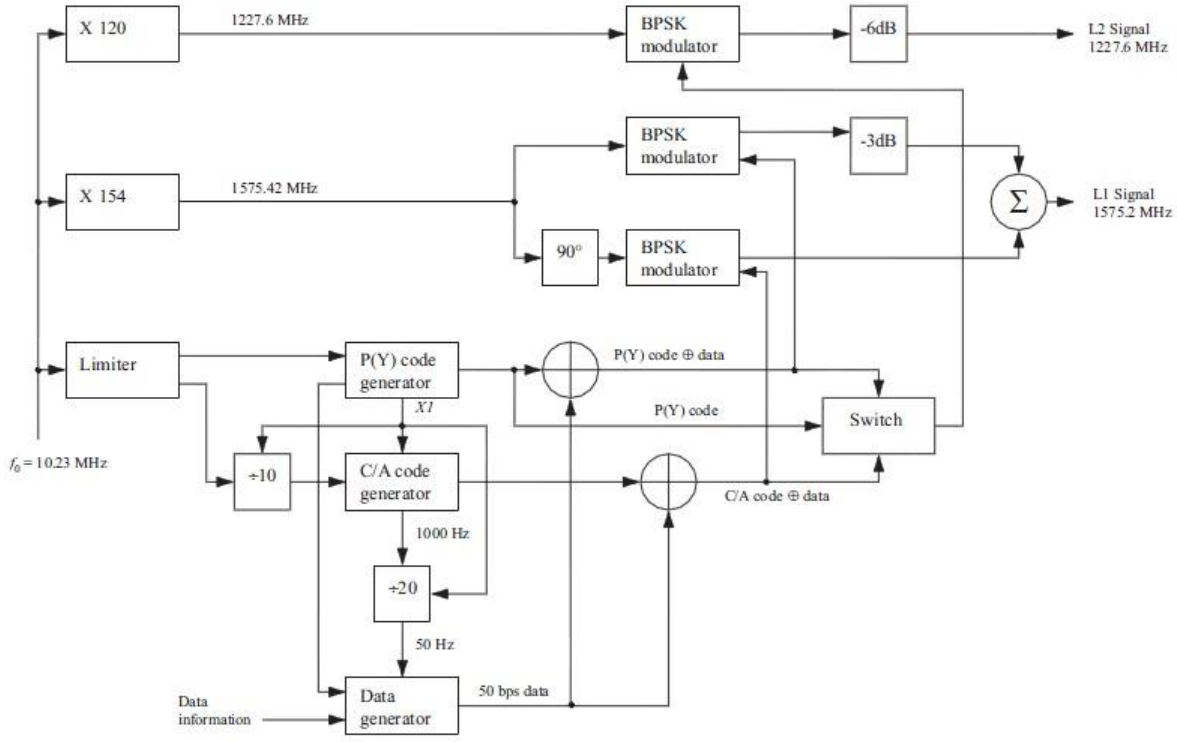


Abbildung 3: Erzeugung des GPS-Signals im Satelliten (Quelle: [7])

Für eine ausreichende Positionsbestimmung über eine Entfernungsmessung müssen immer vier Satelliten im Sichtfeld des Empfängers sein [7] [8].

2.3 GNSS-Empfänger

GNSS-Empfänger stellen das Nutzersegment für GNSS dar. Neben Produkten für den Massenmarkt, welche auf einem einzelnen Chip in fast allen handelsüblichen Geräten verbaut sind, zum Beispiel in einem Smartphone, gibt es High-End Produkte für eine hochpräzise Positionsbestimmung. Seit einiger Zeit gibt es auch GNSS-Empfänger, welche vollständig in die Software implementiert sind. Diese bieten den Vorteil, dass sich Änderungen und Erweiterungen der Funktionsweise durch Umprogrammieren des Softwarecodes vornehmen lassen. Um einen GNSS-Softwareempfänger betreiben zu können, muss ein Front-End bereitgestellt werden, welches die Verbindungsschnittstelle zwischen der analogen und digitalen Welt herstellt. Anstelle von spezifizierten Front-Ends für GNSS lassen sich dafür auch SDR-Systeme verwenden, die dafür noch angepasst werden müssen.

2.3.1 Funktionsweise von GNSS-Empfängern

Um aus dem an der Erdoberfläche ankommenden, sehr schwachen GNSS-Signal die Navigationsdaten für die Positionsbestimmung zu ermitteln, ist eine ganze Reihe von Schritten im Empfänger notwendig. Für eine erfolgreiche Positionsbestimmung des Empfängers müssen vier Satelliten in Sichtweite sein. Das erste Glied in einem GNSS-Empfänger ist ein rauscharmer HF-Verstärker, welcher das schwache, unter dem Rauschlevel liegende Signal verstärkt. Das verstärkte Signal wird anschließend mit einem Superheterodynempfänger heruntergemischt und mit einem Analog-Digital-Umsetzer (engl.: Analog-to-Digital-Converter; Abk.: ADC) quantisiert. Diese arbeiten in handelsüblichen GNSS-Empfängern mit einer Auflösung von 1-2 Bit. Auch wird in einigen Anwendungsfällen mit einer Auflösung von bis zu 8 Bit gearbeitet. Die Signalauswertung der empfangenen Signale wird mithilfe der diskreten Korrelationsfunktion vorgenommen.

2.3.2 Störsignalunterdrückung in GNSS-Empfängern

In heutzutage erhältlichen GNSS-Empfängern sind bereits Methoden zur Störsignalerkennung und Unterdrückung in ersten Versuchen vorhanden. Drei verschiedene Geräte wurden dazu genauer auf deren Funktion untersucht. Es wurden zwei GNSS-Empfänger der Firma NAVILOCK® und ein High-End GNSS-Empfänger der Firma Septentrio verwendet. Tabelle 2 zeigt die Auflistung dieser Geräte mit entsprechenden Spezifikationen.

Tabelle 2: Untersuchte GNSS-Empfänger auf Störsignalunterdrückung und -erkennung

Firma	Model	Chipsatz (Hersteller)	Betriebsart
NAVILOCK®	NL-602U	u-blox 6 (u-blox)	Erkennen
NAVILOCK®	NL-442U	SiRFstarIV™ GSD4e (CSR)	Erkennen + Unterdrücken
Septentrio	PolaRx4TR PRO	Septentrio	Erkennen + Unterdrücken

Die Hauptkomponente in diesen GNSS-Empfängern spielen die eingebauten Chips, mit welchen die Positionsbestimmung erfolgt. Nachfolgend sind alle betrachteten Empfänger mit deren Unterdrückungseigenschaften aufgelistet. Heutzutage ist es nur möglich schwache kontinuierliche Schwingungen mittels Filter und Pulse Blanking zu unterdrücken. Dies sind die ersten Anfänge in der Störsignalunterdrückung bei GNSS-Empfängern. Die Gründe für den langsam voranschreitenden Entwicklungstand liegen in dem großen zeitlichen Entwicklungsaufwand, dem hohen Energieverbrauch von Empfängern in der Störsignalunterdrückung und der geringen Nachfrage am Markt. Durch eine mögliche Zunahme an sogenannten Jammern (engl.: personal privacy devices; PPDs) kann letztere steigen.

Der u-blox 6 SuperSense® Chipsatz ist im GPS- und Galileo-Empfängermodul NL-602U der Firma NAVILOCK® eingebaut [9]. In diesem ist die Unterdrückung von kontinuierlichen Störsignalen bis zu einer Leistung von 25 dB stärker als in einem konventionellen GPS-Empfänger [10].

Der SiRFstarIV™ GSDE3 ist der aktuelle Chipsatz von CSR und findet Verwendung im GPS-Empfänger NL-442U der Firma NAVILOCK® [11]. Dieser verfügt über eine Erkennung und Unterdrückung von Störungen. Laut Datenblatt unterdrückt dieser in-band Jammer bis zu 80 dB-Hz und kann bis zu 8 CW Jammer erkennen [12].

Der PolaRX4TR PRO ist ein geodätischer High-End GNSS-Empfänger von Septentrio. Dieser bietet ebenfalls wie die vorherigen Empfänger eine Störsignalunterdrückung durch adaptive Störsignalunterdrückung mithilfe von automatischer Verstärkungssteuerung (engl.: automatic gain control; Abk.: AGC). Dieser Empfänger unterdrückt kontinuierliche Schwingungen und gepulste Störungen [13].

2.4 Störquellen

Da das GNSS-Signal, bis es auf der Erdoberfläche ankommt, ein sehr schwaches Signal ist, ist dieses sehr anfällig gegenüber Störquellen. Die Störquellen können in zwei Gruppen gegliedert werden. Zum einen gibt es unabsichtliche Störquellen, welche aus anderen Systemen mit den GNSS-Signalen kollidieren. Zum anderen existieren auch absichtliche Störquellen, welche dazu verwendet werden, um bewusst die Funktionsweise von GNSS zu beeinträchtigen. Mit beiden wird die Messgenauigkeit der Position negativ beeinflusst, bis hin zum kompletten Positionsverlust. Als unabsichtliche Störquellen gelten unter anderem das Radar (Radio Detection and Ranging), welches in Abbildung 4 dargestellt ist, und das DME (Distance Measuring Equipment) für die Entfernungsfeststellung eines Luftfahrzeuges. Beide Systeme nutzen den gleichen Frequenzbereich wie GPS und Galileo. Neben diesen gibt es noch die harmonische Störung, welche zum Beispiel von TV- oder Radiosendern ausgehen können. Zum Beispiel besitzen schmale Frequenzbänder innerhalb des FM-Bandes (87,5 – 108 MHz) harmonische Oberwellen, welche das GNSS-Signal belasten. Die Kanäle bei 107,9 und 105,1 MHz weisen die 15te harmonische in der Nähe von GPS und Galileo auf. Absichtliche Störquellen werden aufgrund der immer weiter verbreiteten Nutzung von GNSS für die Navigation durch gezielte Störattacken

herbeigeführt. Dazu werden sogenannte „Jammer“ eingesetzt [51]. Solche Jammer können günstig im Internet erworben oder auch selbst gebaut werden. Oft können diese mit der Zigarettenbuchse im Auto und ohne spezielle Kenntnisse betrieben und genutzt werden. Der Besitz solcher Jammer ist in Deutschland erlaubt (Recht auf Eigentum), jedoch nicht der Betrieb [14] [15]. Diese Jammer senden verschiedene Störsignale aus, angefangen von einer einfachen kontinuierlichen Schwingung bis hin zu komplexen Störsignalen. Das Erkennen und Unterdrücken solcher Störsignale ist ein aktuelles und wichtiges Forschungsthema.

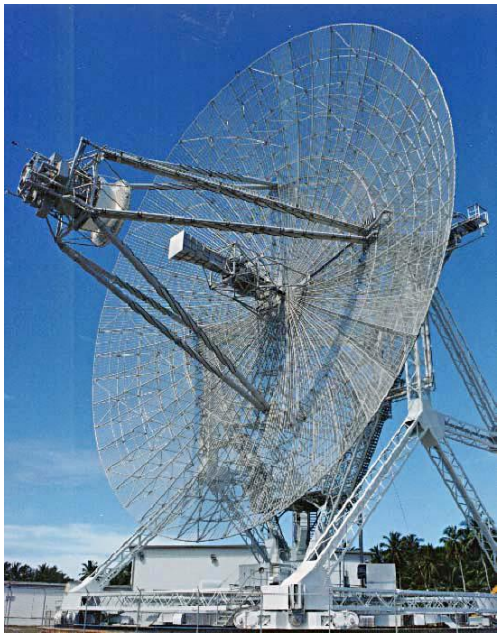


Abbildung 4: Radarantenne (Quelle: [16])

Abbildung 5: GNSS-Jammer (Quelle: [17])

Neben den Störungen im Frequenzband von GNSS gibt es noch andere Einflüsse: Fehler im Raum-, Kontroll- und Benutzersegment, Signalausbreitungsfehler durch ionosphärische und troposphärische Refraktion und die Mehrwegeausbreitung durch Reflektion der Signale.

3 Präzisieren der Aufgabenstellung

Anhand der dargestellten theoretischen Grundlagen wird nun die Aufgabenstellung für die Realisierung dieser Diplomarbeit präzisiert. Diese besteht aus dem Aufzeichnen von störbehafteten GNSS-Signalen, der Unterdrückung von Störsignalen und dem Realisieren einer GNSS-Störsignalunterdrückungseinheit.

3.1 Aufzeichnen von störbehafteten GNSS-Signalen

Um die auftretenden GNSS-Störsignale auf die verschiedenen Algorithmen zur Störsignalunterdrückung anzuwenden, müssen diese reproduzierbar als digitale Samples bereitliegen. Die Aufzeichnung soll unter Einhaltung der projektspezifischen Eigenschaften mit dem am Institut zur Verfügung stehenden GNSS-Softwareempfänger erfolgen, welcher die störbehafteten Satellitensingale als Rohdaten nach dem ADC auf der Festplatte abspeichert. Um eine fehlerfreie Aufzeichnung zu gewährleisten, müssen im ersten Schritt die Datenblätter der verwendeten HF-Geräte studiert werden, um einen fehlerfreien Betrieb dieser zu garantieren. Um eine Reproduzierbarkeit zu gewährleisten, wird anstelle von realen Signalen auf die Verwendung eines GNSS-Simulators zurückgegriffen, womit immer dieselben Bedingungen in der Satellitenkonstellation vorhanden sind. Ebenso geschieht dies mit den Störquellen. Dazu wird ein extra für das ESA-Forschungsvorhaben entwickelter Interferenzen-Generator verwendet, mit welchem die erzeugten GNSS-Signale anschließend komponiert werden. Anhand von technischen Spezifikationen des Auftraggebers werden die verschiedenen Szenarien mit den im Labor zur Verfügung stehenden Mitteln unter Einhaltung von technischen Spezifikationen aufgezeichnet. Der genaue Versuchsaufbau mit den verwendeten Geräten und Komponenten wird dokumentiert. Ebenfalls gilt es den genauen Ablauf bei der Aufzeichnung festzuhalten.

3.2 Unterdrücken von Störsignalen

Nachdem die digitalen Rohdaten, welche Samples genannt werden, bereitstehen, kann mit der Testphase begonnen werden. Die verwendeten Algorithmen zur Störsignalunterdrückung im GNSS-Bereich wurden alle in einem anderen Projekt entwickelt. Die gesamte Auswertung erfolgt mit dem im Projekt erstellten TERMINATE Development Kit (TDK), mit welchem die gewünschten Samples und Algorithmen auswählbar sind. Wegen der Reproduzierbarkeit erfolgt dies im Post-Processing. Bevor eine Unterdrückung der im GNSS-Signal enthaltenen Störung erfolgen kann, muss diese erkannt werden. Jeder Algorithmus muss eine derartige Funktion beinhalten. Anschließend gilt es, die Qualität der Störsignalunterdrückung zu bestimmen, was den Hauptteil der Auswertungen ausmacht. Anhand eines zuvor festgelegten Ablaufplanes werden die ausgewählten störbehafteten GNSS-Signale mit den verschiedenen Algorithmen bearbeitet und liefern ein Ergebnis hinsichtlich der Qualität der Störsignalunterdrückung. Bei den Algorithmen können verschiedene Parameter eingestellt werden, wobei es gilt, die besten

in Abhängigkeit der Art des Störsignals zu ermitteln. Für diesen Vorgang stehen am Institut eine Reihe von Workstations mit Hochleistungsprozessoren bereit. Abschließend werden die jeweiligen Algorithmen gegenübergestellt, um deren Unterschiede darzustellen und eine Aussage über deren Rentabilität in Bezug auf den Unterdrückungserfolg und die Echtzeitverarbeitung zu geben.

3.3 GNSS-Störsignalunterdrückungseinheit

Im letzten Kapitel wird auf einem SDR-System mit integriertem FPGA eine GNSS-Störsignalunterdrückungseinheit realisiert. Dazu wird das USRP-2952R (Universal Software Radio Peripheral) von National Instruments gewählt. Dieses verfügt über einen HF-Transceiver, mit welchem das Empfangen und Senden hochfrequenter Signale möglich ist. Neben dem Transceiver ist ein frei programmierbarer FPGA vorhanden, auf welchem Algorithmen zur digitalen Signalverarbeitung implementierbar sind. In diesem Anwendungsfall wird ein Algorithmus zur Störsignalunterdrückung eingesetzt. Die Programmierung des USRPs, einschließlich des FPGAs, erfolgt mit LabVIEW und LabVIEW FPGA, einer grafischen Programmiersprache. Aufgrund der enthaltenen Empfangs- und Sendeeinheit des USRPs soll die darauf entwickelte GNSS-Störsignalunterdrückungseinheit über zwei Betriebsarten verfügen. Die eine stellt neben dem kommerziellen Front-End die Verwendung als eigenes Front-End mit Störsignalunterdrückung für den am Institut verfügbaren Softwareempfänger dar. Die andere Betriebsart kann das empfangene und bearbeite Signal über die Sendeeinheit wieder ausgegeben und somit als GNSS-Repeater dienen. Das ausgegebene Signal kann an einen beliebigen GNSS-Empfänger weitergegeben werden. Die Steuerung und Änderung wichtiger Parameter des USRPs erfolgt über einen Rechner. Ebenfalls sollen wichtige Daten darstellbar sein. Das empfange Signal soll im Zeit- und Frequenzbereich sichtbar sein, um direkt am Front-End erste Eigenschaften zu überprüfen. Es wird das von National Instruments verwendete Beispielprogramm für das USRP verwendet und mit den benötigten Eigenschaften modifiziert und angepasst.

4 Aufzeichnen störbehafteter GNSS-Signale

Dieses Kapitel befasst sich mit der Aufzeichnung von störbehafteten GNSS-Signalen. Bevor damit begonnen wird, erfolgt eine Erklärung der Vorgehensweise. Anschließend wird auf das Laborsetup eingegangen, mit einer Erläuterung der verwendeten Hardware und Software. Letztendlich werden die störbehafteten GNSS-Signale mit einem abschließenden Ergebnis aufgezeichnet.

4.1 Vorgehensweise

Um die entwickelten Algorithmen auf ihre Funktionsweise zu überprüfen, ist es zunächst notwendig, dass die erforderlichen GNSS-Signale unter Störeinfluss als digitale Roh-Daten zur Verfügung stehen. Da die Wirkung der verschiedenen Algorithmen mit den einstellbaren Parametern immer auf dieselbe Satellitenkonstellation und dieselbe Störung angewendet werden soll, wird jedes Szenario mit der gewünschten Störeigenschaft einmal aufgezeichnet und abgespeichert. Die Aufzeichnungen erfolgen mit dem am Institut vorhandenen Softwareempfänger, welcher zusätzlich über eine Aufnahmefunktion verfügt. Somit herrschen immer dieselben Bedingungen und es ist sichergestellt, dass bei den Auswertungen der verschiedenen Algorithmen immer die gleiche Ausgangssituation gegeben ist.

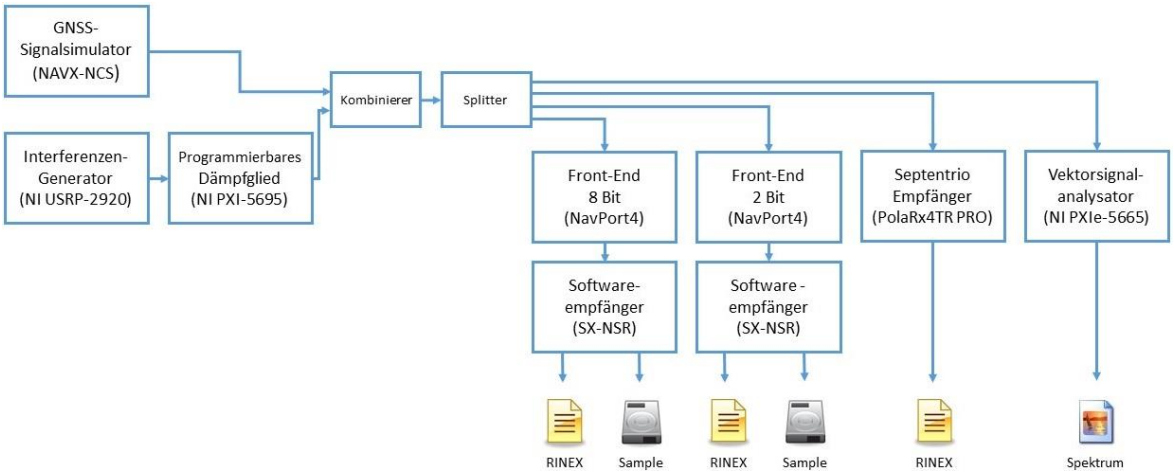


Abbildung 6: Versuchsaufbau für das Aufzeichnen

Der grundlegende Versuchsaufbau für das Aufzeichnen der Szenarien ist in Abbildung 6 dargestellt. Das erste Glied in der Aufzeichnung stellt einen GNSS-Simulator dar, welcher die reproduzierbaren GNSS-Signale erzeugt. Im Vergleich zur Verwendung realer GNSS-Signale hat dies den Vorteil, dass bei dem gewünschten Signal zu jeder Zeit dieselben Bedingungen vorliegen. Somit wird jedes Szenario mit der gleichen Konstellation und Sendeleistung der Satelliten aufgezeichnet. Außerdem werden bei den Versuchen auch Galileo-Signale mit einbezogen, welche über reale Satelliten noch nicht ausreichend vorhanden sind (Stand: 16.01.2015). Das Störsignal wird im Labor mit einem Interferenzen-

Generator (IG) erzeugt. Dies hat ebenfalls den Vorteil, dass anstelle der Verwendung eines echten Jammers oder einer mit einem Signalgenerator erzeugten Störquelle immer reproduzierbare Störsignale mit denselben Parametern und Leistungen erzeugbar sind. Der IG ist mit verschiedenen Szenarien vorkonfiguriert. In Tabelle 3 ist Szenario #1 - #10 dargestellt. Diese Szenarien von Störsignalen sind in einem Skript mit den spezifizierten Eigenschaften im Betriebssystem des Steuerrechners des IG anpassbar. Für die Störung in Szenario #10, mit einer Bandbreite von 12 MHz, wurde ein realer Jammer verwendet. Die Sendeantenne des Jammers 8 lässt sich einfach abmontieren, wodurch das ausgesendete Störsignal direkt über ein HF-Kabel dem nachfolgenden Glied in der Aufzeichnungskette zur Verfügung gestellt werden kann.

Tabelle 3: Festgelegte Störsignale

Szenarios	**Parameter der Störsignale**
#1	Single CW @ 7.5 MHz
#2	Two CWs @ 7.5 MHz and @ 3.5 MHz
#3	Four CWs @ 7.5 MHz and @ 6.5 MHz and @ 3.5 MHz and @ 2.0 MHz
#4	Filtered AWGN with B_{start} = 5 MHz and B_{stop} = 6 MHz
#5	Filtered AWGN centred on f_{FN} = 2 MHz and B= 100 kHz
#6	DME with FWHM = 3.5 µsec, Δt = 12 µsec and PRF = 2000 Hz
#7	Filtered AWGN centred on f_{FN} = 2 MHz with B = 100 kHz + Single CW @ 7.5 MHz
#8	Radar Chirp, Pulsed Period = 3 msec, Pulse Duration = 1 µsec, Max Freq. = 0.33 MHz
#9a	Chirp Signal, Central Frequency = 5.5 MHz, Max Freq. = 6 MHz, Rise Time = 5 µsec, Fall Time = 1 µsec
#9b	Changes of the sweep time to 60 µsec: Rise Time = 50 µsec, Fall Time = 10 µsec
#9c	Changes of the sweep time to 600 µsec: Rise Time = 500 µsec, Fall Time = 100 µsec
#10	Jammer 8: BW = 12 MHz [50]

Nach dem IG ist ein programmierbareres Dämpfglied geschaltet, um die Leistung während der Aufzeichnung kontinuierlich in der Dämpfung zu verringern. Am Anfang wird das Störsignal mit einem zuvor bestimmten Referenzwert gedämpft, bis das Störsignal so stark unterdrückt wird, dass kein Einfluss mehr auf das Träger-Rausch-Verhältnis (C/N_0) der GNSS-Signale bestand. Das C/N_0 gibt in den Tests und Auswertungen Auskunft über die Signalqualität des GNSS-Signals. Im zeitlichen Verlauf der Aufzeichnung wird die Dämpfung immer niedriger, was eine immer stärker werdende Störung zur Folge hat. Das GNSS-Signal und das mit einem verstellbaren Dämpfglied vorgeschaltete Störsignal werden anschließend mit einem Kombinierer zusammengefügt. Das Ergebnis ist das störbehaftete GNSS-Signal, mit welchem die Algorithmen auf ihre Wirksamkeit in Bezug

auf den Erkennungs- und Unterdrückungserfolg und auf ihre Qualität zur Echtzeitfähigkeit getestet und ausgewertet werden.

Mit einem Splitter wird das störbehaftete GNSS-Signal mehreren Empfangseinheiten bereitgestellt. Diese umfassen einen Softwareempfänger mit einem Front-End von 8 Bit Quantisierung, einen Softwareempfänger mit einem Front-End von 2 Bit Quantisierung sowie einen geodätischen GNSS-Empfänger und einen Vektorsignalanalysator zur Leistungsbestimmung. Bei der Leistungsbestimmung wird die höchste Leistung im Spektrum mit der „Peak-Search-Funktion“ bestimmt. Zusätzlich wird die Durchschnittsleistung der gesamten Bandbreite bestimmt. Mit den beiden Softwareempfängern wird das Signal aufgenommen und eine RINEX-Datei (Receiver Independent Exchange Format) erzeugt. Der geodätische GNSS-Empfänger liefert ebenfalls eine RINEX-Datei, welche als Referenzwert zu den beiden Softwareempfängern dient.

4.2 Laborsetup

In diesem Kapitel wird die verwendete Hardware und Software aufgelistet und beschrieben. Darüber hinaus werden die Dateiformate erläutert.

4.2.1 Verwendete Hardware und Software

Nachfolgend erfolgt eine Auflistung der für die Aufzeichnungen verwendeten Hardware und Software. Diese werden näher vorgestellt und erläutert.

4.2.1.1 GNSS-Signalsimulator (NavX-NCS)

Der GNSS-Signalsimulator NavX-NCS Professional (Navigation Constellation Simulator) der Firma IFEN GmbH wird für das Simulieren von GNSS-Signalen verwendet. Dieser Simulator ist ein Multifrequenz-Signalgenerator, mit welchem sich je nach Konfiguration die folgenden Signale auf bis zu 108 Kanäle simulieren lassen: GPS L1/L2/L5, Galileo E1/E5ab/E6, GLONASS G1/G2, BeiDou B1/B2, QZSS L1 und SBAS L1/L5. Zusätzlich kann dieser mit einer intern verbauten Rauschquelle ausgestattet werden, was das Einstellen eines definierten Grundrauschen erlaubt. Die Steuerung erfolgt mittels des NCS Control Center, das auf einem beliebigen Rechner mit Windows als Betriebssystem lauffähig ist. Die Schnittschelle des Simulators zum Steuerrechner erfolgt mittels Ethernet [18]. Es wird empfohlen, den Simulator direkt mit dem Steuerrechner zu verbinden, um die volle Datenrate des Ethernet Anschlusses zu gewährleisten.

Abbildung 7: NavX®-NCS Professional – GNSS-Signalsimulator (Quelle: [18])

4.2.1.2 Interferenzen-Generator (USRP-2920)

Der Interferenzen-Generator (IG) wurde von der Firma WISER für das ESA-Projekt [49] auf einer SDR-Plattform realisiert. Dieser kann auf dem NI USRP-2920 oder Ettus N210 betrieben werden. Für die Aufzeichnung im Labor des Institutes stand das NI USRP-2920 zur Verfügung. Diese preisgünstige Lernplattform für den Laboreinsatz in der HF-Technik verfügt über eine durchstimmbare Mittenfrequenz von 50 MHz - 2,2 GHz. Der interne Verstärker bei der Sendeeinheit ist in 1 dB Schritten von 0 dB – 31 dB veränderbar. Bei einer Samplebreite von 16 Bit beträgt die durchschnittliche Echtzeitbandbreite 20 MHz mit einer Abtastrate von 25 MS/s. Die Verbindung mit dem Steuerrechner erfolgt über die Ethernet Schnittstelle [19]. Der IG kann bis zu sechs verschiedene Arten von Störsignalen erzeugen und beinhaltet eine, zwei und vier kontinuierliche Schwingungen (engl.: continuous wave; Abk: CW), schmal- und breitbandige Rauschstörungen, DME, Radar und Chirp Signale (Simulation von Jammer) als Störung. Die Software ist lauffähig unter dem Linux Betriebssystem Ubuntu.

Abbildung 8: NI USRP-2920 genutzt für den Interferenzen-Generator (Quelle: [19])

4.2.1.3 Programmierbares Dämpfglied (PXI-5695)

Damit die Leistung während der Aufzeichnung von störbehafteten GNSS-Signalen stetig im gleichen Intervall und Pegel geändert werden kann, wird ein programmierbares Dämpfglied nach dem IG geschaltet. Bei dem NI PXI-5695 handelt es sich um ein programmierbares HF-Dämpferglied mit zwei Kanälen von 50 MHz bis 8 GHz. Der erste Kanal weist eine fest vorgegebene Dämpfung von 28,5 dB auf. Mit dem zweiten Kanal kann die Dämpfung in 0,5 dB Schritten von 0 – 31,5 dB mittels Software programmiert und in definierten Schritten automatisch mit einer gewünschten Rampenfunktion geändert werden. In Reihe können beide Kanäle eine Gesamtdämpfung von bis zu 60 dB erreichen [20].

Abbildung 9: NI PXI-5695 – Programmierbares Dämpfglied (Quelle: [20])

4.2.1.4 Vektorsignalanalysator (PXIe-5665)

Der Hochleistungs-Vektorsignalanalysator NI PXIe-5665 ist ein modularer, dreistufiger Superheterodyn-Analysator und wird als Vektorsignalanalysator (VSA) verwendet. Dieses HF-Messgerät bietet einen Frequenzbereich von 20 Hz – 3,6/14 GHz bei einer Bandbreite von 25 MHz oder 50 MHz. Der Vektorsignalanalysator weist ein sehr niedriges Phasenrauschen von -129 dBc/Hz, mittlere Rauschpegel von -165 dBm/Hz und eine Amplitudengenauigkeit von ± 0,10 dB auf [21]. Die Steuerung erfolgt mit der beiliegenden Software NI-RFSA Soft Front Panel über einen Rechner mit Windows als Betriebssystem.

Abbildung 10: NI PXIe-5665 – Vektorsignalanalysator (Quelle: [21])

4.2.1.5 GNSS-Softwareempfänger (SX-NSR)

Der GNSS-Softwareempfänger SX-NSR von der Firma IFEN GmbH ist ein vollständig in Software implementierter GNSS-Empfänger. Mit dem Betrieb eines Front-Ends kann dieser zur Echtzeitverarbeitung von GNSS-Signalen verwendet werden. Zusätzlich ist der Softwareempfänger im Post-Processing Modus betreibbar, was einen großen Vorteil bei einem in Software implementiertem Empfänger darstellt. Dabei werden digitalisierte Samples von einem Speichermedium eingelesen und verarbeitet. Diese Funktion bietet ebenfalls den Vorteil, dass die Samples beliebig bearbeitet werden können, wie zum

Beispiel nach dem Anwenden von Algorithmen zur Störsignalunterdrückung. Für das Aufzeichnen und Auswerten der störbehafteten GNSS-Signale wird Version 2.6 des GNSS-Softwareempfängers verwendet [22].

4.2.1.6 Front-End (NavPort-4)

Ein Front-End stellt eine Verbindung von der analogen zur digitalen Welt her. Die Einstellungen der Parameter des Front-Ends können direkt aus dem Softwareempfänger vorgenommen werden. Für die Aufzeichnung wird das für den Softwareempfänger verfügbare NavPort-4 Front-End verwendet. Dieses arbeitet mit einer Abtastrate von 20,48 MHz, um die Bandbreite von Galileo E1 mit 10,24 MHz abdecken zu können. Die maximale Auflösung bei der Quantisierung beträgt 2 Bit. Um eine höhere Signaldynamik zu erreichen, arbeitet in diesem Projekt das Front-End mit einer Auflösung von 8 Bit. Die Freischaltung von 8 Bit erfolgte speziell dafür.

Abbildung 11: NavPort-4 – Front-End (Quelle: [22])

4.2.1.7 GNSS-Empfänger (Septentrio PolaRxTR PRO)

Der Septentrio PolaRxTR PRO ist ein High-End GNSS-Hardwareempfänger für GPS-, GLONASS-, Galileo- und BeiDou-Signale und dient neben dem Softwareempfänger als Referenzempfänger für das Aufzeichnen der Samples. Die Ansteuerung kann mittels COM, USB oder Ethernet erfolgen. Da der GNSS-Empfänger über einen integrierten Webserver verfügt, kann dieser, neben der dafür vorgesehenen Steuersoftware - RX-Control, über einen Internet-Browser eingestellt und angesteuert werden [13]. Das Loggen der RINEX-Dateien erfolgt auf dem internen Speicher, welche zu einem späteren Zeitpunkt über den eigenen FTP-Server des Empfängers heruntergeladen werden können.

Abbildung 12: Septentrio PolaRxTR PRO – GNSS-Empfänger (Quelle: [13])

4.2.2 Dateiformate

Während der Aufzeichnungs- und Testphase wird mit verschiedenen Dateiformaten gearbeitet. Auch in den hervorgegangenen Kapiteln ist häufig die Abkürzung RINEX aufgetreten. Um diese Begrifflichkeit besser zu verstehen, erfolgt an dieser Stelle eine kurze Erläuterung. Neben dem RINEX-Format wird noch das NMEA-Format erklärt.

4.2.2.1 RINEX-Format

Das RINEX-Format (Receiver Independent Exchange Format) ist ein empfängerunabhängiges Datenspeicher- und Datenaustauschformat. Es wurde von dem Astronomischen Institut der Universität Bern für den einfachen Austausch von GPS-Daten entwickelt. Die hintergründigen Aspekte waren, dass die meisten geodätischen Verarbeitungsprogramme eine fest definierte Reihe von Messgrößen verwenden. Diese sind die Trägerphasenmessung, Pseudorangemessung (Code) und die Beobachtungszeit. In der Regel übernimmt die Software die Beobachtungszeit für eine gültige Phasen- und Codemessung für alle beobachteten Satelliten. Die in der Datei enthaltenen Informationen werden für Verarbeitungsprogramme benötigt. Anfangs war es mit Version 1.0 nur möglich, GPS-Signale im RINEX-Format darzustellen. Mit der Weiterentwicklung bis hin zu Version 3.02 kamen neben GPS noch weitere GNSS-Systeme hinzu [23]. In Abbildung 13 ist der Auszug des Inhaltes einer RINEX-Datei dargestellt, welche der Softwareempfänger verwendet und ausgibt. Es handelt sich dabei um die Version 3.01. Am Anfang der Datei befindet sich eine Reihe von Informationen über die verwendete Hardware und Software, die mit der Zeile „END OF HEADER“ abgeschlossen wird. Danach werden die Werte der GNSS-Daten für jeden einzelnen Satelliten geloggt und fortlaufend in diese Datei geschrieben. Die erste Spalte gibt den Satelliten an, wobei der erste Buchstabe das jeweilige GNSS identifiziert (G = GPS, E = Galileo), gefolgt von zwei Ziffern, die den genauen Satelliten anhand der PRN (bei GLONASS heißt die PRN „slot number“) zeigen. In den folgenden Zeilen sind alle spezifischen GNSS-Daten der Satelliten gespeichert. Die Dateiendung der RINEX-Datei ist „.13o“.

```
     3.01           OBSERVATION DATA    M                   RINEX VERSION / TYPE
G = GPS  R = GLONASS  E = GALILEO  S = GEO                  COMMENT
C = COMPASS  M = MIXED                                      COMMENT
SX-NSR              IFEN GMBH           20131106 152529 GPS PGM / RUN BY / DATE
EXECUTABLE BUILD ON:    Aug 12 2014 at:       17:13:13      COMMENT
NSR_MARKER                                                  MARKER NAME
0000                                                        MARKER NUMBER
SOMEONE             USER                                    OBSERVER / AGENCY
R0001               IFEN SX_NSR_RT_800  2.6.0               REC # / TYPE / VERS
A0001               UNKNOWN                                 ANT # / TYPE
NO VALID RECEIVER POSITION -> MARKER SET TO EQUATOR         COMMENT
  6378000.0000        0.0000        0.0000                  APPROX POSITION XYZ
        0.0000        0.0000        0.0000                  ANTENNA: DELTA H/E/N
G   16 C1C L1C D1C S1C C2X L2X D2X S2X C2D L2D D2D S2D C5X  SYS / # / OBS TYPES
       L5X D5X S5X
E   20 C1X L1X D1X S1X C5X L5X D5X S5X C7X L7X D7X S7X C8X  SYS / # / OBS TYPES
       L8X D8X S8X C6X L6X D6X S6X
R    8 C1C L1C D1C S1C C2C L2C D2C S2C                      SYS / # / OBS TYPES
C   12 C2I L2I D2I S2I C7I L7I D7I S7I C6I L6I D6I S6I      SYS / # / OBS TYPES
S    4 C1C L1C D1C S1C                                      SYS / # / OBS TYPES
     1.000                                                  INTERVAL
DBHZ                                                        SIGNAL STRENGTH UNIT
  2013    11     6    15    25   28.9991938     GPS         TIME OF FIRST OBS
                                                            END OF HEADER
> 2013 11 06 15 25 28.9991938  0 10      -0.000806084376
G23  23914991.283 7        53.07317     -3277.936 7        44.091           0.000           0.000
G02  20973097.984 7        52.29117       -25.907 7        44.082           0.000           0.000
G07  24670414.158 7        53.15217       505.004 7        44.069           0.000           0.000
G17  23402481.495 7        52.48417      2903.047 7        44.043           0.000           0.000
G18         0.000           0.000      -2283.111 1      -100.000           0.000           0.000
E26  26794283.861 7    141808.00017     -2636.051 7        44.409           0.000           0.000
E17  24728270.603 7    105056.00017      -170.364 7        44.395           0.000           0.000
E23  26912701.224 7   -105920.00017      1908.927 7        44.345           0.000           0.000
E06  25654779.450 7    -61880.00017      1455.054 7        44.342           0.000           0.000
E10  27824446.467 7     78624.00017     -3752.480 7        44.323           0.000           0.000
> 2013 11 06 15 25 29.9991940  0 10      -0.000805877907
G17  23401928.774 7     -2850.611 7      2902.751 7        44.121           0.000           0.000
G07  24670316.901 7      -451.551 7       504.107 7        44.099           0.000           0.000
```

Abbildung 13: Auszug des Inhalts einer RINEX-Datei

4.2.2.2 NMEA

Die NMEA (National Marine Electronics Association) ist eine Vereinigung, welche sich für die Ausbildung und den Fortschritt in der Marine Elektronik engagiert. Bei dem NMEA-Protokoll handelt es sich um einen Standard für die Kommunikation zwischen Navigationsgeräten auf Schiffen. Ebenso lässt sich dieser für die Kommunikation zwischen GPS-Empfängern und PCs sowie mobilen Endgeräten nutzen [24].

4.3 Einstellungen der Hardware

Vor den Aufzeichnungen wird die verwendete Hardware und Software eingestellt. Die Einstellungen und Kalibrierungen erfolgen anhand gewünschter Spezifikationen des Auftraggebers für das Projekt.

4.3.1 Einstellungen des GNSS-Signalsimulators

Die Leistung der ausgesendeten GNSS-Signale wird so gewählt, dass bei dem Hauptempfänger (Softwareempfänger mit einem Front-End von 8 Bit Quantisierung) ein Träger-zu-Rausch-Verhältnis (C/N_0) von 44,5 dB-HZ bei GPS und 45 dB-Hz bei Galileo vorhanden ist. Tabelle 4 zeigt die ermittelten Einstellungen des GNSS-Simulators.

Tabelle 4: Parameter des GNSS-Simulators

Parameter	Wert
Power Level Density	-150 dBm/Hz
Galileo E1 Power	-95 dBm
GPS L1 Power	-101 dBm

4.3.2 Einstellungen des Interferenzen-Generators

Der Interferenzen-Generator (IG) bietet die Möglichkeit, verschiedene Arten von Störsignalen zu erzeugen und über die Transceiver-Einheit auszugeben. Als Steuerrechner dient hierbei ein leistungsfähiger Laptop mit dem Betriebssystem Linux, der mit einem SSH-Client über die Ferne gesteuert werden kann. Da der IG von einem Projektpartner auf dem USRP von Ettus realisiert wurde, musste das NI USRP noch mit der Firmware und dem FPGA-Image von Ettus versehen werden. Dies erfolgte mit der beiliegenden Software zum Aufspielen der Firmware. Für jedes Szenario ist ein individuell anpassbares Skript auf dem Steuerrechner vorhanden, mit welchem die verschiedenen Parameter der Störsignale anhand der ESA-Anforderungen [49], dargestellt in Tabelle 3, angepasst werden. Im weiteren Verlauf der Aufzeichnung müssen keine zusätzlichen Einstellungen vorgenommen werden. Ein Starten und Stoppen des IG über die Linux-Konsole reicht aus.

4.3.3 Einstellungen des Softwareempfängers

Die Aussteuerung der Amplitude bei der Digitalisierung der GNSS-Signale sollte einen Wert von 10 Digits aufweisen. Dazu wird die AGC-Funktion des Front-Ends abgeschaltet und mit einem festen Wert bestimmt. Die Einstellung aller Parameter für das Front-End erfolgten über den Softwareempfänger. Der benötigte AGC-Wert für den Hauptempfänger in Abhängigkeit vom GNSS-Signalsimulator ist in Tabelle 5 dargestellt.

Tabelle 5: Eingestellter AGC-Wert des Front-Ends

Parameter	Wert
AGC-Wert (8 Bit Front-End)	14

4.3.4 Einstellungen des Hardwareempfängers

Der Septentrio GNSS-Empfänger wird mit den vorkonfigurierten Grundeinstellungen betrieben. Als einzige Änderung wird das Loggen der RINEX-Dateien im 12-Stunden-Modus mit allen Satelliten aktiviert. Diese lassen sich mittels der Steuersoftware RX-Control oder bei Anschluss über Ethernet mit dem Browser von dem internen Speicher des Empfängers herunterladen und auf dem lokalen PC abspeichern. Die benötigen Satelliten können in der RINEX-Datei individuell ausgelesen werden. Der AGC-Wert des Empfängers passt sich automatisch an die Eingangsleistung an.

4.3.5 Einstellungen des programmierbaren Dämpfgliedes

Damit die Leistung während der Aufzeichnung von störbehafteten GNSS-Signalen stetig im gleichen Intervall und Pegel geändert werden kann, wird ein programmierbares Dämpfglied nach den IG geschaltet. Für das programmierbare Dämpfglied muss die Referenzdämpfung bestimmt werden, bei der keine Degradierung von C/N_0 auftritt.

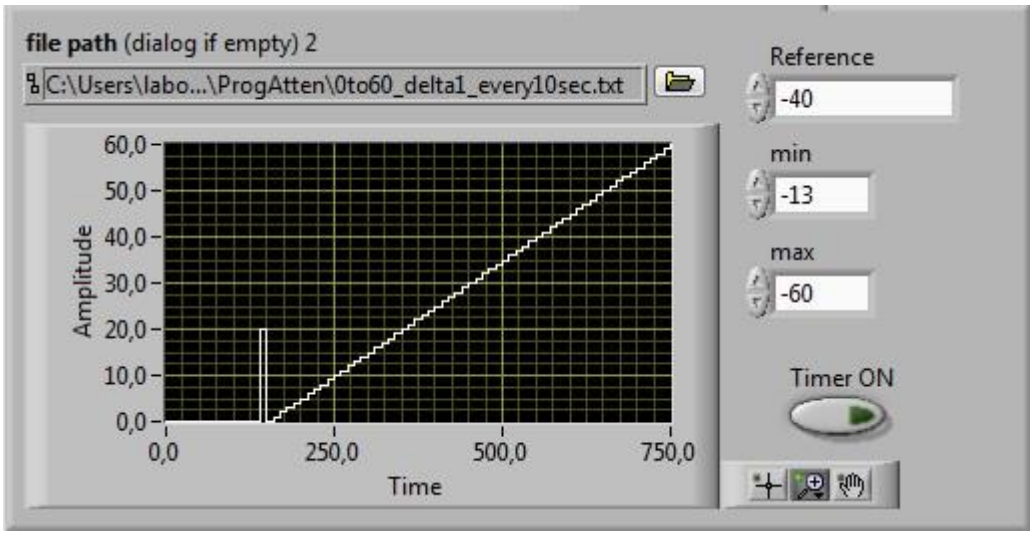


Abbildung 14: Zeitlicher Verlauf der Dämpfung

Die mit LabVIEW programmierte Software zum Steuern des programmierbaren Dämpfgliedes bewirkt, dass die Dämpfung alle 10 s um 1 dB herabgeht. Die Rampenfunktion ist in Abbildung 14 dargestellt. Die Minderung der Dämpfung hat eine stärkere Auswirkung der Störung auf das GNSS-Signal zur Folge. Somit wird das GNSS-Signal im zeitlichen Verlauf einer immer stärker werdenden Störung ausgesetzt, bis hin zur vollständigen Degradierung des GNSS-Signals.

4.4 RINEX-Viewer

In diesem Kapitel erfolgt die Erläuterung des Programmes, mit welchem die RINEX-Dateien ausgewertet werden. Dieses wurde im Zuge der Arbeit um eine zusätzliche Funktion erweitert, damit das gleichzeitige Einlesen von mehreren RINEX-Dateien möglich ist.

4.4.1 Erläuterung

Der RINEX-Viewer ist ein am Institut entwickeltes Programm zum Einlesen und Auswerten der vom Empfänger erzeugten RINEX-Dateien. Unter anderem ist es mit diesem Tool möglich die Plots zu erzeugen, welche das Signal-Rausch-Verhältnis (C/N_0) der GNSS-Signale in Abhängigkeit der Leistung des Störsignals anzeigen. Dieses wird benötigt, um die erforderlichen Plots zu erzeugen, mit welchen eine Aussage über die Qualität der GNSS-Signale getroffen werden kann.

4.4.2 Erweiterung

Da mit dem RINEX-Viewer immer nur eine RINEX-Datei ausgewählt und eingelesen werden kann, wurde dieses um ein zusätzliches Programm erweitert, welches das jeweilige Hauptprogramm mehrfach aufruft. Somit ist das Einlesen mehrere RINEX-Dateien auf einmal möglich. Dazu muss ein entsprechendes Skript erzeugt werden, das sich auf die jeweiligen RINEX-Dateien mit den gewählten Satelliten bezieht. Die verschiedenen Algorithmen zur Unterdrückung der Störsignale sollen zum Vergleich in einem Plot darstellbar sein. Abbildung 15 zeigt die Bedienkonsole, in welcher das gewünschte, zuvor erzeugte Skript, eingelesen wird. Im oberen Feld „file path (RINEX-Script)“ wird das Skript, ausgewählt. Die folgenden Ausgabefelder zeigen die momentan eingelesenen Werte. Sollte mit einer gerade eingelesen RINEX-Dateien etwas nicht in Ordnung sein, stoppt das Programm automatisch an der entsprechenden Stelle, damit eine Überprüfung erfolgen kann.

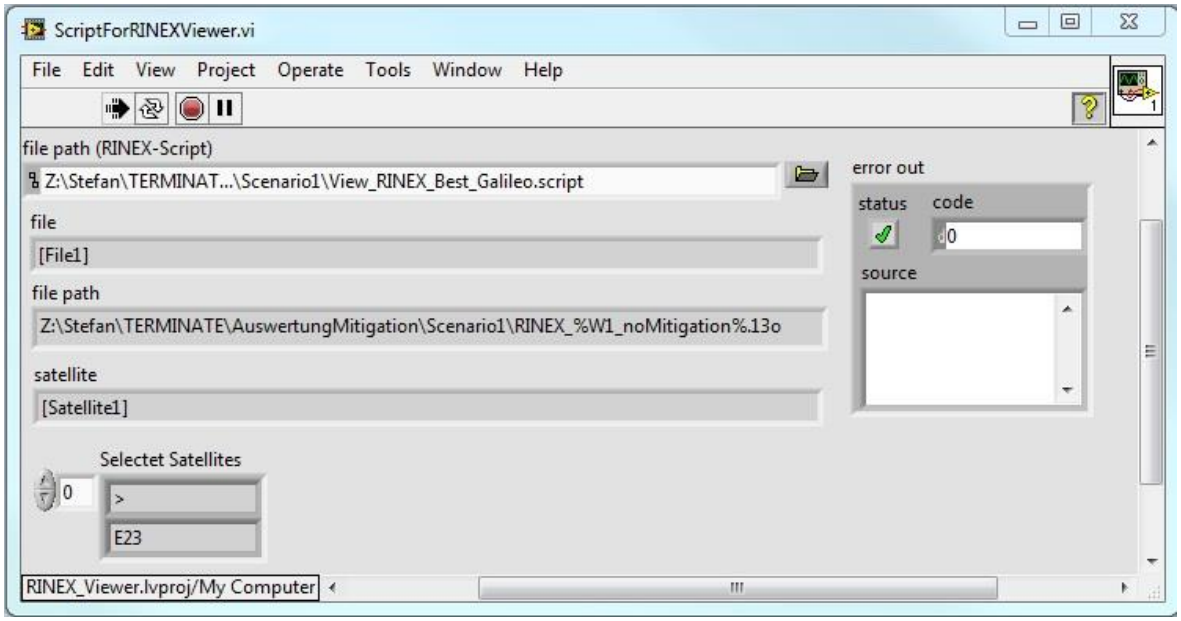


Abbildung 15: Ansicht des Programmes „ScriptForRINEXViewer“

In Abbildung 16 ist ein Skript für den RINEX-Viewer dargestellt, welches mit einem beliebigen Editor angepasst werden kann. Die in den eckigen Klammern enthaltenen Werte dienen lediglich der Übersicht und werden nicht an den RINEX-Viewer übergeben. Als erstes werden der Pfad und der Dateiname eingelesen, gefolgt von dem ausgewählten Satelliten. Das Skript kann theoretisch bis zu einer beliebigen Länge fortgeführt werden, allerdings empfiehlt es sich, um eine bessere Übersicht in den erzeugen Plots zu erreichen, nur eine bestimmte Anzahl an RINEX-Dateien einzulesen und darzustellen.

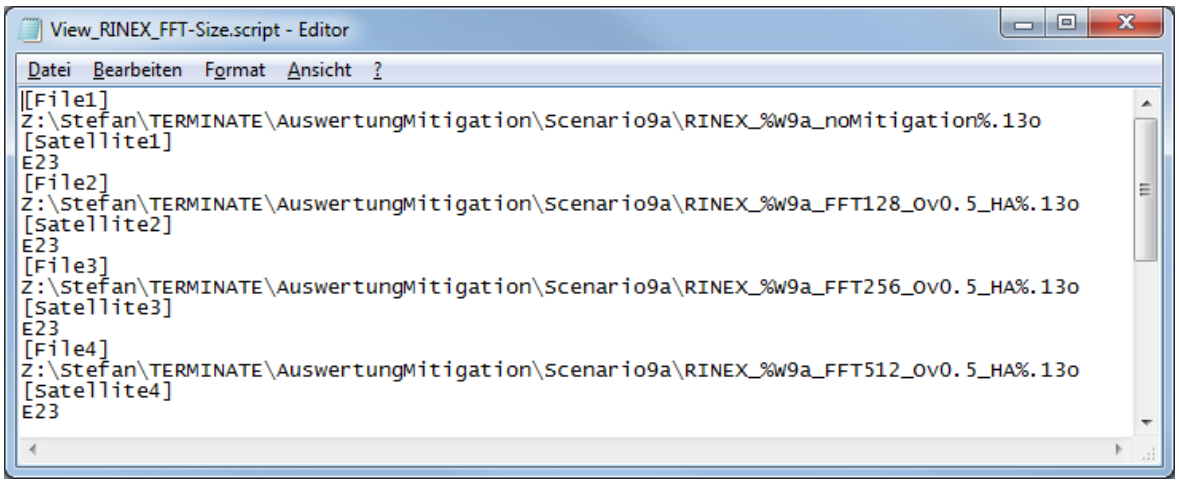


Abbildung 16: Skript des RINEX-Viewer

4.5 Aufzeichnen der störbehafteten GNSS-Signale

Mittels der zuvor gezeigten Hardware und Software erfolgten die Aufzeichnungen der jeweiligen Szenarien im Serverraum des Institutes. Hierbei wird eine konstante Umgebungstemperatur von 21 °C gewährleistet.

Nach erfolgter Aufzeichnung der störbehafteten GNSS-Signale stehen folgende Dateien zur Verfügung: Mit den drei Empfängern wird jeweils eine RINEX-Datei erzeugt. Die Samples werden mit den zwei Softwareempfängern aufgenommen, wodurch am Ende zwei Samples mit unterschiedlicher Quantisierung zur Verfügung stehen. Ein Front-End arbeitet wie im Projekt vorgegeben mit einer Quantisierung von 8 Bit, das Zweite mit einer Quantisierung von 2 Bit. Die 2 Bit Quantisierung wurde optional aufgenommen. Dies ist nicht Bestandteil der weiterführenden Untersuchungen, sondern dient lediglich als Zusatz, falls weitere Untersuchungen in nachfolgenden Arbeiten gemacht werden wollen. Die Abtastrate bei beiden Front-Ends beträgt 20,48 MHz. Das Sample mit 8 Bit Quantisierung wird anschließend auf die Algorithmen angewendet.

4.5.1 Vorgehensweise bei der Aufzeichnung

Bei der Aufzeichnung müssen alle Geräte für die Aufzeichnungen in einem bestimmten Ablauf für die Reproduzierbarkeit gestartet werden. Um dies zu gewährleisten, wurde zuvor ein definierter Ablaufplan für die Reihenfolge erstellt und festgelegt. Diese ist in Tabelle 6 dargestellt.

Tabelle 6: Vorgehensweise bei der Aufzeichnung

Schritt	Beschreibung
0	Alle alten RINEX-Dateien und Samples aus dem Speicherordner löschen
1	GNSS-Signalsimulator + Septentrio starten
2	Warten auf fixe Position des Septentrio Empfängers
3	Starten der RINEX-Aufzeichnung des Septentrio
4	Beide Softwareempfänger starten (RINEX-Aufzeichnung erfolgt automatisch)
5	Zwei Minuten warten
6	Aufzeichnung der Signale beider Softwareempfänger + Skript für programmierbares Dämpfglied starten
7	90 s warten, anschließend den Interferenzen-Generator starten
8	Bei Endmeldung des Skriptes im Dämpfglied, 10 s warten bevor alles gestoppt wird

Sobald die Aufzeichnung abgeschlossen ist, ist bei dem Hauptempfänger ein digitalisiertes GNSS-Signal mit einer Größe von etwa 12 GByte vorhanden. Die Größe des digitalisierten GNSS-Signals ist abhängig von der Abtastrate, der Quantisierung und der Aufzeichnungsdauer. Bei einer Abtastrate von 20,48 MHz, einer Quantisierung von 8 Bit und einer Aufzeichnungsdauer von 10 Minuten ergibt sich eine Größe von 12,288 GByte:

$$Samplegr\ddot{o}\ss e\ [Byte] = Abtastrate\ [Hz] * Zeit[s]$$

4.5.2 Durchführen der Aufzeichnungen

Sind alle verwendeten Geräte eingestellt und kalibriert und der fest definierte Ablauf festgelegt, kann mit dem Durchführen der Aufzeichnungen begonnen werden. Wie zuvor angegeben erfolgten die Aufzeichnungen von Szenario #1 - #9c mit dem IG und dem zuvor spezifizierten Versuchsaufbau. In Szenario #10 wurde das störbehaftete GNSS-Signal anstelle des IG mit einem realen Jammer erzeugt. Es ist nötig, diese Phasenmodulation von 6 MHz Bandbreite mit einem Vektorsignalgenerator VSG zu erzeugen oder auf die Verwendung eines echten Jammers zurückzugreifen. Um in den Versuchen auf einen realen Jammer zurückzugreifen, wurde ein solcher verwendet. Jammer 8 [50] am Institut bietet für Szenario #10 die benötigten Eigenschaften. Bevor dieser benutzt wurde, musste noch die Ausgangsleistung eingestellt werden. Auf dem Gehäuse befindet sich ein Potentiometer, mit welchem die Leistung einstellbar ist. Um eine ideale Sendeleistung des Jammers zu bestimmen, wurde als erstes zum Schutz der Hardware mit der niedrigsten Leistung gesendet und zusätzlich das programmierbare Dämpfglied mit maximaler Dämpfung eingestellt. Anschließend wurde die Leistung des Jammers so dimensioniert, dass die gewünschte Sendeleistung vorhanden ist. In Abbildung 17 ist der Wert des Potentiometers des Jammers dargestellt.

Abbildung 17: Wert des Potentiometers von Jammer 8

Nach der Aufzeichnung der Szenarien liegen je Szenario drei verschiedene RINEX-Dateien vor, mit welchen die C/N_0-Werte mit dem RINEX-Viewer ausgelesen und dargestellt wurden.

In Abbildung 18 ist der mit dem RINEX-Viewer ausgewertete Plot von den RINEX-Dateien aus Szenario #1 dargestellt. Es ist erkennbar, dass die Qualität der GNSS-Signale signifikant mit steigender Leistung der Störung abnimmt. Betrachtet wurde hierbei das Galileo-Signal von Satellit Nummer 6. Während die vertikale Achse Auskunft über die Qualität des GNSS-Signals mit dem Indikator C/N_0 gibt, bestimmt die horizontale Achse die Leistung des Störsignals. Die Auswertung der RINEX-Datei des geodätischen GNSS-Empfängers Septentrio ist in schwarz dargestellt, grün der des Softwareempfängers mit dem Front-End von 2 Bit Quantisierung und rot der Softwareempfänger mit 8 Bit Quantisierung. Bei dem Front-End mit 8 Bit Quantisierung wurde bei der Aufzeichnung von Galileo-Signalen am Anfang ein besseres Ergebnis des C/N_0 erzielt. Auf eine Darstellung der entstandenen Plots von Szenario #2 - #10 wurde verzichtet, da diese ein sehr ähnliches Ergebnis liefern.

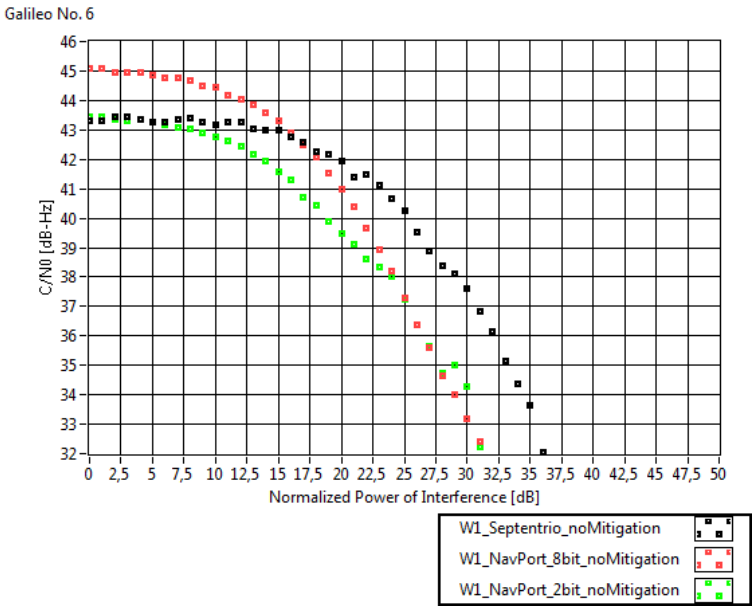


Abbildung 18: C/N_0 von Galileo Satellit Nr.6 von den verschiedenen Empfängern mit der Störung von Szenario #1

4.6 Ergebnis

Der Softwareempfänger mit einer Quantisierung von 8 Bit liefert ohne Störungen in allen Aufzeichnungen die beste Signalqualität. Dieser weist im Plot aus Abbildung 18 ein um über 1 dB besseres C/N_0 auf. Wie in Abbildung 19 ersichtlich, bewirkt eine höhere Quantisierung bei der Digitalisierung von GNSS-Signalen eine geringere Degradierung des C/N_0. Bei dem Eintreten einer über die Zeit stärker werdenden Störung gleicht sich jedoch die Qualität der Signale mit dem Softwareempfänger von 2 Bit Quantisierung an. Der Septentrio-Empfänger besitzt bei stärker werdender Störung ein besseres C/N_0, was auf das AGC zurückzuführen ist.

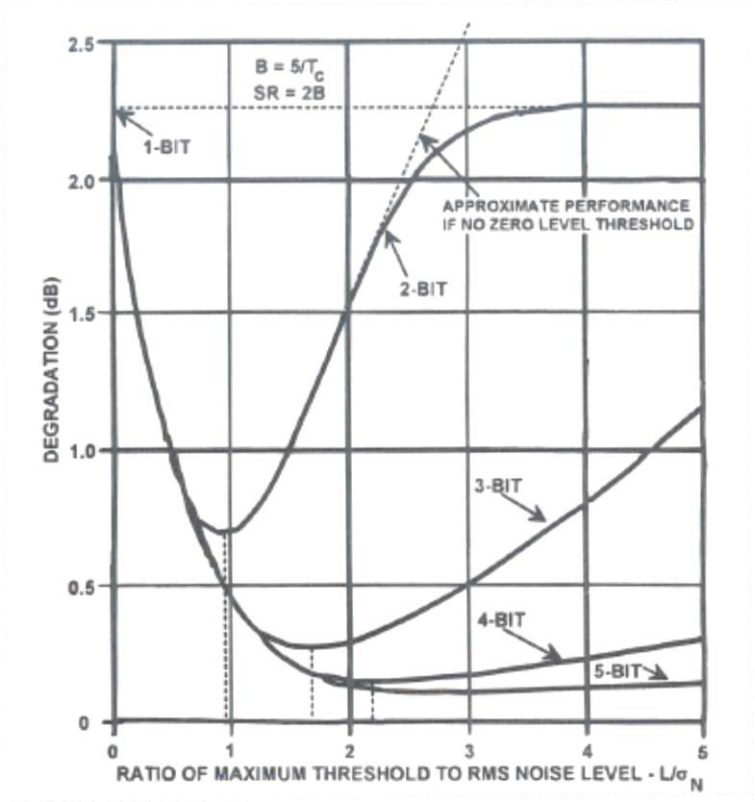


Abbildung 19: Wirkung auf das C/N_0 bei unterschiedlicher Quantisierung (Quelle: [52])

Diese aufgezeichneten, als Sample vorliegenden GNSS-Signale unter Störsignaleinfluss werden nun mit den verschiedenen zur Verfügung stehenden Algorithmen auf die Erkennung, Unterdrückung und auf die Qualität der Echtzeitfähigkeit untersucht.

5 Unterdrückung von Störsignalen

In diesem Kapitel wird auf das Unterdrücken von Störsignalen eingegangen. Bevor eine Unterdrückung von Störsignalen erfolgen kann, müssen diese erkannt werden. Anhand des zur Verfügung stehenden TERMINATE Development Kit (TDK) werden die vorliegenden Algorithmen auf die jeweiligen Szenarien angewendet und der effizienteste ermittelt. Sind die Samples prozessiert, werden diese anschließend mit dem Softwareempfänger eingelesen, um eine Aussage über die Signalqualität zu erhalten. Die Qualität eines jeden Algorithmus für das Szenario wird grafisch in einem Plot zur Übersicht dargestellt. Für jedes Szenario wird ein Plot mit dem jeweils besten Algorithmus und dem des „rohen“ störbehaften GNSS-Signals erzeugt. Nach einer kurzen Einleitung des Programmes zur Anwendung der Algorithmen auf die Samples, wird auf die in dieser Diplomarbeitet genauer betrachteten Störsignale und Algorithmen eingegangen. Danach erfolgte eine abschließende Auswertung hinsichtlich des Erfolgs der Unterdrückung sowie eine Einschätzung zur Eignung der Algorithmen. Zusätzlich wurde noch die Echtzeitfähigkeit der Algorithmen mit dem TDK bewertet.

5.1 Erkennen von Störsignalen

Um störbehaftete GNSS-Signale zu unterdrücken, muss zuvor eine Erkennung von Störsignalen stattfinden. Störsignale können durch Beobachtung des Träger-zu-Rauschverhältnisses (C/N_0), Leistungsmessung oder durch die Analyse der Korrelationswerte detektiert werden. Ist ein Störsignal vorhanden, erhöht sich die Rauschleistung. Tritt eine Übersteuerung des ADC auf, führt dies zu einer Degradierung der Korrelation [25].

Mit den aufgezeigten Methoden kann zwar erkannt werden, ob eine Störung vorliegt, jedoch nicht, um welche Art von Interferenz es sich handelt. Möchte man das Störsignal charakterisieren, muss direkt nach dem ADC eine Charakterisierung im Zeit- oder Frequenzbereich erfolgen [25].

Für das Erkennen einer Störung ist es ausreichend, definierte Intervalle in einer gewissen Zeitperiode zu betrachten.

5.2 TERMINATE Development Kit

Mit dem zur Verfügung stehenden Programm „TERMINATE Development Kit (TDK)“ erfolgen die Auswertungen auf ein mit Interferenzen behaftetes GNSS-Signal in Bezug auf die Erkennung, Unterdrückung und Echtzeitfähigkeit von CPU-gestützten Algorithmen zur Störsignalunterdrückung. Dieses Programm wurde am Institut entwickelt. Es bietet die Möglichkeit, die aufgezeichneten störbehafteten GNSS-Signale einzulesen, anschließend den gewünschten Algorithmus zur Störsignalunterdrückung auszuwählen und wieder als „bereinigtes“ Sample abzuspeichern. Zudem lässt sich das Spektrum des Eingangs- und

Ausgangssignal für den direkten Vergleich im TDK grafisch anzeigen. Da am Anfang aufgrund der Vorgehensweise der Aufzeichnung noch keine Störung sichtbar ist, besitzt das TDK zusätzlich die Möglichkeit, einen Offset festzulegen. Mit diesem Offset kann der Beginn, ab welchem Zeitpunkt die Abtastwerte eingelesen werden, festgelegt werden. Das Arbeiten mit dem Offset empfiehlt sich, da am Anfang noch keine Interferenz sichtbar ist und noch keine Aussage über den Unterdrückungserfolg möglich ist. Bevor das Sample am ganzen Stück durch das TDK bearbeitet wird, kann man einen definierten Zeitpunkt nach vorne gehen und im Fenster des Spektrums einen eventuell anfallenden Erfolg direkt erkennen. Ist dieser gegeben, kann mit dem vollständigen Testlauf begonnen werden.

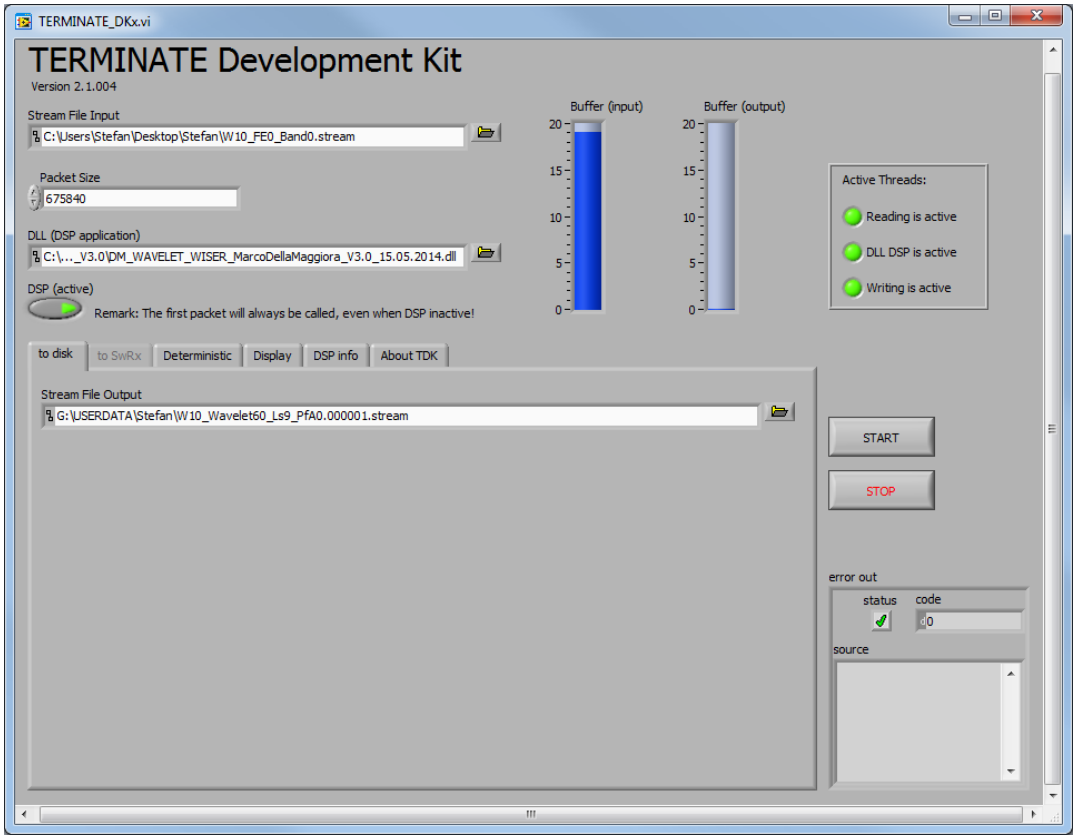


Abbildung 20: TERMINATE Development Kit

Im Feld „Stream File Input“ wird der jeweilige, zuvor aufgezeichnete Sample ausgewählt. Die jeweiligen DLL Dateien, in welchen die Algorithmen enthalten sind, bieten noch eine zugehörige Text-Datei, in welcher die Parameter für die Algorithmen eingestellt werden können. Außerdem wird eine Ausgangsdatei erstellt, in der Informationen über das erkannte Störsignal enthalten sind. Im Reiter „to disk“ wird der Pfad festgelegt, in welchem das Sample nach der Störsignalunterdrückung gespeichert wird. Der Name der Datei enthält am Anfang den Namen des jeweiligen Szenarios, gefolgt von dem Namen des Algorithmus und den Werten der jeweiligen Parameter. Somit ist später sichtbar, mit welchen Parametern der Algorithmus auf das einzelne Szenario angewendet wird.

5.3 Verwendete Störsignale

Die vorliegende Arbeit bezieht sich auf vier verschiedene Arten von Störsignalen. Diese wurden neu nummeriert. Gewählt wurden Störsignale mit folgenden Eigenschaften: Vier kontinuierliche Schwingungen, Rauschen, Chirp-Signal und ein gepulstes Signal. Somit sind die Arten der häufig vorkommenden Störungen abgedeckt. Die Mittenfrequenz des GNSS-Signals liegt bei 5,5 MHz und die Abtastrate f_{sample} beträgt bei der Aufzeichnung 20,48 MHz. Tabelle 7 zeigt eine Übersicht der verwendeten Störsignale und deren Eigenschaften. Wie in Kapitel 4 dargestellt, wurden diese bereits im Labor erzeugt und aufgenommen.

Tabelle 7: Verwendete Störsignale und deren Eigenschaften

Nummer	Störsignal	Eigenschaften
1	Vier kontinuierliche Schwingungen	@ 7,5 MHz @ 6,5 MHz @ 3,5 MHz @ 2,0 MHz
2	Gefiltertes Rauschen	Mittenfrequenz = 2 MHz B = 100 kHz
3	Chirp-Signal	Mittenfrequenz = 5,5 MHz Max. Frequenz = 6 MHz Anstiegszeit = 5 µs Fallzeit = 1 µs
4	Radar Chirp	Pulsperiode = 3 ms Pulsdauer = 1 µs Max. Frequenz = 0,33 MHz

In den untergeordneten Kapiteln finden detaillierte Beschreibungen der Störsignale mit einer grafischen Darstellung im Zeit- und Frequenzbereich statt. Für das Erstellen der Grafiken wurde mit LabVIEW ein „SampleViewer“ programmiert, mit welchem die Samples eingelesen und dargestellt werden können. Zusätzlich ist dieses Programm mit einem Offset ausgestattet, um das Signal ab einem beliebigen Zeitpunkt einzulesen. Bei der grafischen Darstellung der Signale variiert die Samplelänge im Zeitbereich von jeder verschiedenen Interferenz, um diese im besten sichtbaren Bereich darzustellen. Zum Vergleich ist in Abbildung 21 ein aufgezeichnetes GNSS-Signal ohne Störeinfluss im Zeit- und Frequenzbereich dargestellt.

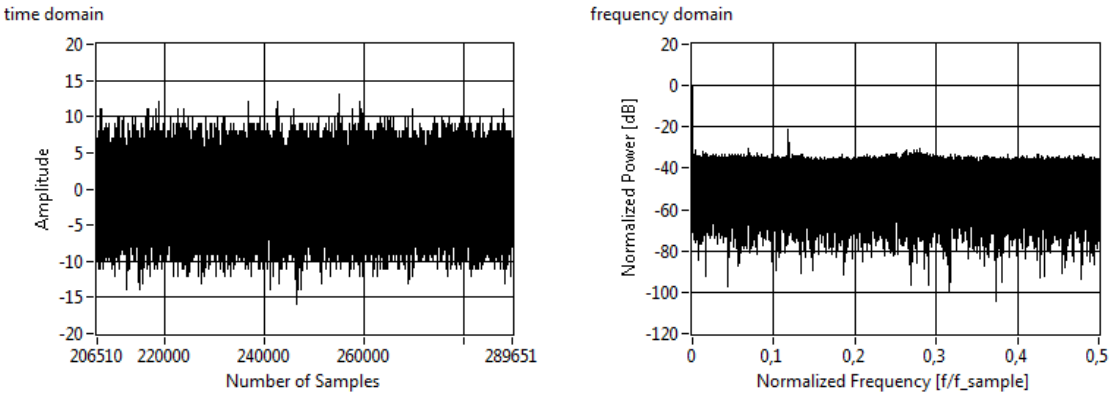


Abbildung 21: GNSS-Signal im Zeit- und Frequenzbereich ohne Interferenz

Die Aussteuerung der Amplitude von 10 Digits ist wie in den Spezifikationen vorgegeben aus dem Zeitbereich sichtbar. Das GNSS-Signal wird als Rauschen wahrgenommen, bedingt dadurch, dass das GNSS-Signal unterhalb des Rauschpegels liegt. Im Frequenzbereich ist bei einer normalisierten Frequenz von 0,268 eine minimale Anhebung im Frequenzbereich sichtbar, welche das Spektrum des GNSS-Signals zeigt.

Bei einer normalisierten Frequenz von etwa 0,12 ist im Spektrum des GNSS-Signals eine Spitze von rund 10 dB über den Rauschlevel zu erkennen. Hierbei handelt es sich um einen Artefakt, welcher bei der gesamten Signalkette auftreten kann. Beginnend von den Signalgeneratoren (GNSS-Signalsimulator und IG) über die analogen Komponenten bis hin zur Digitalisierung der GNSS-Signale. Dieser feine Artefakt ist sichtbar, da sich die Auflösung im Zeit- und Frequenzbereich aufgrund der hohen Anzahl an dargestellten Abtastwerten (675.840) erhöht. Der darin schwache Energieanteil hat nur einen marginalen Einfluss auf das GNSS-Signal.

5.3.1 Vier kontinuierliche Schwingungen (Szenario #1)

Bei dem ersten verwendeten Störsignal handelt es sich um vier kontinuierliche Schwingungen (CW), mit welchen das GNSS-Signal gestört wurde. Ein störbehaftetes GNSS-Signal mit einer Schwingung als Störung wurde in diversen Forschungsvorhaben betrachtet [26], weshalb sich hierbei auf vier Schwingungen unterschiedlicher Frequenzen bezogen wurde. Diese Art von Störung beeinträchtigt GNSS deutlich, weil das Frequenzspektrum von einem GNSS-Signal mit vier CW stark belastet wird. Das CW-Signal kann aus einem Monoton- oder aus AM/FM-Signal bestehen. Ausgehen können diese von Amateurradio- oder TV-Sender bei Fehlfunktionen.

Abbildung 22 zeigt das GNSS-Signal mit der Störkomponente im Zeit- und Frequenzbereich. Im Zeitbereich ist deutlich zu erkennen, dass mit Auftreten der Störung die Amplitude des Signals steigt. Dies ist auf eine Erhöhung der enthaltenen Leistung im gesamten Signal zurückzuführen. Die zusätzliche Leistung geht von dem Störsignal aus. Auch ist die Überlagerung eines CW im enthaltenen Signal sichtbar. Zum Erkennen der weiteren darin enthaltenen CW ist ein Vergrößern beziehungsweise Verkleinern des angezeigten Bereichs der X-Achse nötig, da es sich um mehrere CW mit unterschiedlichen Frequenzen handelt. Im Frequenzbereich sind die vier Spitzen von den jeweiligen CW mit

einer normalisierten Leistung von 5 dB sichtbar. Der Rauschlevel, in welchem das GNSS-Signal enthalten ist, liegt bei einer normalisierten Leistung von -40 dBm.

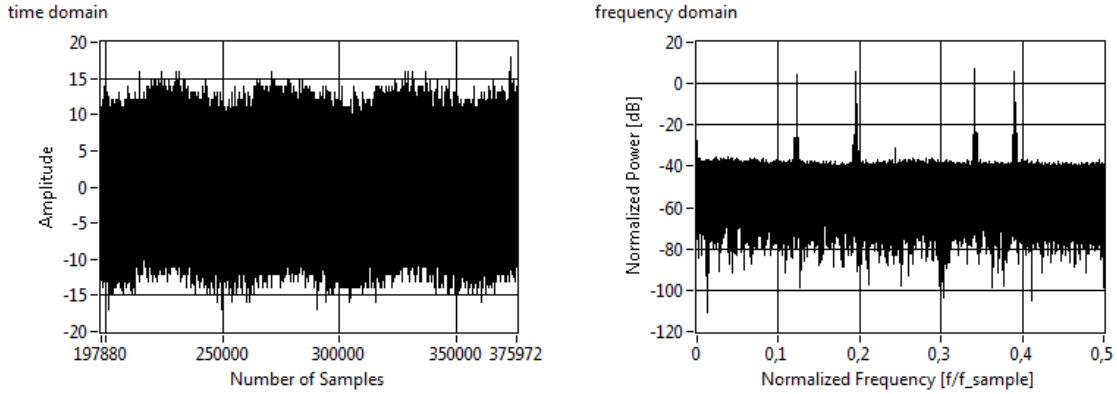


Abbildung 22: GNSS-Signal im Zeit- und Frequenzbereich mit vier kontinuierlichen Schwindungen

5.3.2 Gefiltertes Rauschen (Szenario #2)

Eine häufige Störkomponente stellt Rauschen dar. Rauschen kann nicht nur durch äußere Einwirkungen auftreten, sondern auch durch Fehlfunktionen im Empfänger und in der kompletten Signalempfangskette. Bei dem Erzeugen dieser Störung wurde im Labor das additive weiße gaußsche Rauschen (engl.: additive white Gaussian noise; Abk.: AWGN) mit einer Bandbreite von 100 kHz bei einer Mittenfrequenz von 2 MHz verwendet.

In Abbildung 23 ist das störbehaftete GNSS-Signal im Zeit- und Frequenzbereich dargestellt. Im Zeitbereich ist eine kontinuierliche Schwingung zu sehen. Im Frequenzbereich ist bei der Mittenfrequenz eine Erhöhung der Leistung mit einer Bandbreite von 100 kHz sichtbar.

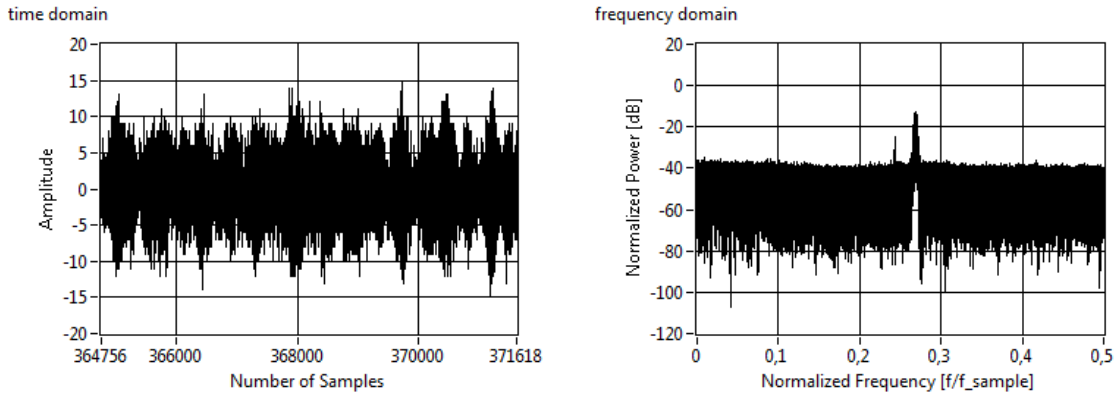


Abbildung 23: GNSS-Signal im Zeit- und Frequenzbereich mit gefiltertem Rauschen

5.3.3 Chirp-Signal (Szenario #3)

Als Chirp-Signal („zwitschern“) bezeichnet man ein Signal, welches sich zeitlich in seiner Frequenz ändert. Dieses wird häufig von Jammern, auch als PPDs [50] (personal privacy devices) bezeichnet, ausgesendet. Das Chirp-Signal ist mit den verwendeten Parametern in Tabelle 8 dargestellt, welches Szenario #9a aus den Definitionen des ESA-Projekts [49]

entspricht. Im Vergleich zu Szenario #9b und #9c weist dieses eine kürzere Durchlaufzeit der Frequenzwechsel auf. Daraus resultiert auch eine kürzere Anstiegs- und Abfallzeit.

Tabelle 8: Parameter des Chirp-Signals

Parameter [Einheit]	Erläuterung	Wert gewähltes Chirp-Signal
max. Frequenz [Hz]	Die maximale Frequenzabweichung Δf des Chirp-Signal.	max. Frequenz = 6 MHz
Anstiegszeit [s]	Die Zeit T_{up} während die Frequenz steigt.	Anstiegszeit = 5 µs
Abfallzeit [s]	Die Zeit T_{dw} während die Frequenz fällt.	Abfallzeit = 1 µs

Abbildung 24 zeigt das störbehaftete GNSS-Signal mit einem Chirp-Signal als Störkomponente im Zeit- und Frequenzbereich. Im Zeitbereich ist eine Änderung der Frequenz im zeitlichen Verlauf erkennbar. Dies ist die typische Eigenschaft des Chirp-Signals. Aufgrund der vielen vorkommenden Frequenzen ist im Frequenzbereich über die gesamte Bandbreite eine große Anzahl an spektralen Spitzen sichtbar.

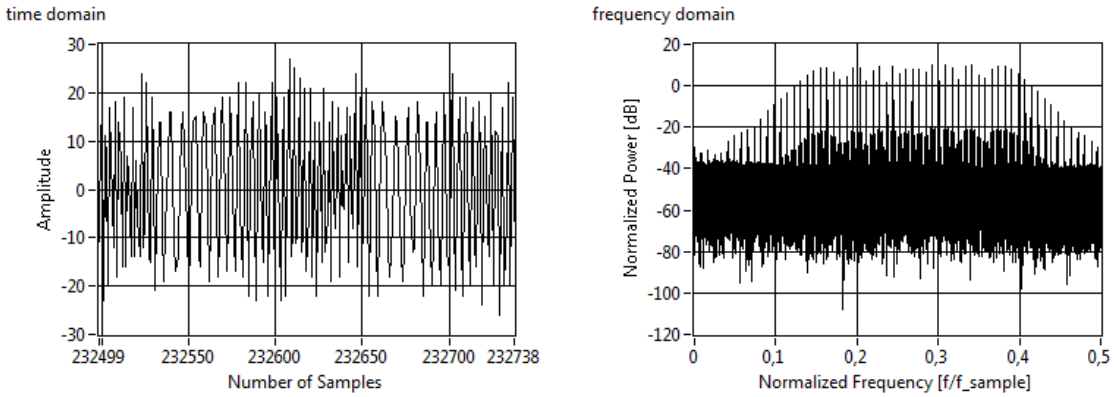


Abbildung 24: GNSS-Signal im Zeit- und Frequenzbereich mit einem Chirp-Signal

5.3.4 Radar (Szenario #4)

Um alle möglichen Arten von auftretenden Störsignalen abzudecken, wird noch das Radar gewählt. In dem Aeronautical Radio Navigation Service (ARNS) Band, wo GPS L5 und Galileo E5a/b Signale übertragen werden, sind viele andere zivile Systeme angesiedelt, wie das DME und Radar. Die GNSS-Signale müssen sich diesen Frequenzbereich mit anderen Diensten teilen. Das Radar (Radio Detection and Ranging) ist ein Gerät zur Erkennung und Entfernungsbestimmung von Gegenständen durch elektromagnetische Wellen. Dieses System strahlt im Frequenzbereich ein gepulstes Signal von 1250-1310 MHz mit einer variablen Leistung aus. Die Pulsdauer ist im Bereich von 1 µs bis 100 µs variabel. Ebenso ist die Pulswiederholung variabel [27]. Zur Untersuchung der Auswirkung von Radarsystemen auf GNSS-Systeme gibt es Veröffentlichungen [28], weshalb bei dieser Störung eine Entwicklung zur Gegenmaßnahme interessant ist. Das Radar sendet somit ein gepulstes Chirp-Signal, welches auf die kreisförmige Aussendung der Radarsignale

zurückzuführen ist. Zur Untersuchung für die Unterdrückungsmöglichkeit eines Radarsignals wurde dieses mit den in Tabelle 9 angegebenen Parametern erzeugt.

Tabelle 9: Parameter des Radars

Parameter [Einheit]	Spezifikation	Wert
max. Frequenz [Hz]	Die maximale Frequenz-abweichung Δf des Radar Chirp	0,33 MHz
Pulsperiode [s]	Die Dauer einer Periode	3 µs
Pulsdauer [s]	Die Dauer eines Pulses	1 µs

In Abbildung 25 ist das störbehaftete GNSS-Signal im Zeit- und Frequenzbereich mit einem Radar dargestellt. Sichtbar sind im Zeitbereich die pulsartigen Ausschläge des Radars. Im Frequenzbereich ist eine normalisierte Leistung von -15 dB des Radars zu erkennen.

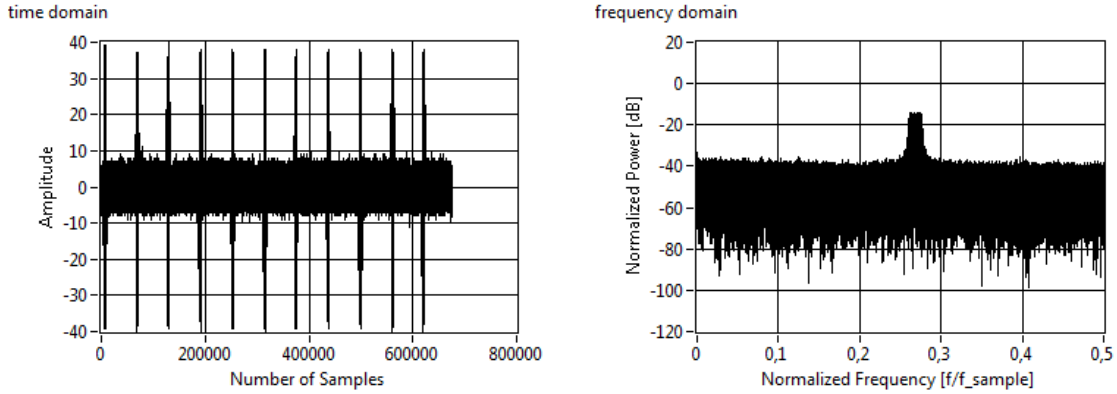


Abbildung 25: GNSS-Signal im Zeit- und Frequenzbereich mit Radar

5.4 Verwendete Algorithmen zur Störsignalunterdrückung

In der digitalen Signalverarbeitung können Daten im Zeitbereich oder auch durch Transformation im Frequenzbereich bearbeitet werden. Als Unterdrückungsmethode im Zeitbereich ist das Pulse Blanking und der Notch Filter zu sehen. Der Pulse Blanking arbeitet allerdings nur bei gepulsten Störsignalen. Im Frequenzbereich ist die Fast Fourier-Transformation (FFT), Karhunen-Loève-Transformation (KLT) und Wavelet-Transformation (WT) zu sehen.

Um die Störsignale aus den GNSS-Signalen zu entfernen, wurden im Rahmen des ESA-Projektes [49] verschiedene Algorithmen implementiert. Um eine Unterdrückung der Störsignale zu realisieren, werden Frequenzanteile, welche das Störsignal beinhalten, herausgefiltert. Sie werden auf Null gesetzt oder dem Rauschlevel angepasst. Da in diesem störbehafteten Teil auch Informationen des GNSS-Signals enthalten sind, wird auch Energie des GNSS-Signals entfernt.

Tabelle 10: Verwendete Algorithmen zur Störsignalunterdrückung

Nummer	Algorithmen	Einstellbare Parameter
1	FFT	FFT size: 2^n (32 – 65536) Overlap Factor: 0, 0.5 or 0.75 Window Type: RE, HA, BH Normalized Threshold: 3.75, 4.0 or 4.3884
2	KLT	Window Length: 32, 64, 128, 256, 512, 1024 bins Window Type: RE, HA, HAM, HFT90D Threshold: -2.0 – 0.0 (optimal below -1.7; by developer)
3	Wavelet	QMF length [N_W]: 60 or 90 Selective tree levels [L_s]: 9 or 11 False alarm probability [Pfa]: 0.01 or 0.000001
4	Pulse Blanking	
5	Notch Filter	False alarm probability [Pfa]: 0.0001 or 0.00001 Contraction factor [kAlpha]: 0.75, 0.85 or 0.9 Number of successive scans [nScans]: n

5.4.1 Schnelle Fourier-Transformation

Der schnelle Fourier-Transformation (engl.: Fast Fourier Transform; Abk.: FFT) ist ein Algorithmus zur effizienten Berechnung der diskreten Fourier-Transformation (engl.: discrete Fourier Transform; Abk.: DFT). Mit der FFT kann ein digitales Signal im Zeitbereich in seine Frequenzanteile zerlegt werden [29]. Der im FFT-Algorithmus enthaltene Erkennungs- und Unterdrückungsprozess basiert auf einer anpassbaren Schwellwertbestimmung. Eine bestimmte Anzahl von Samples im Zeitbereich wird zunächst mit einer gewählten Fenstermethode multipliziert. Danach wird eine FFT auf dem gefensterten Signal durchgeführt und der Modulo wird für jede FFT-Komponente errechnet. Pins, welche über dem Schwellwert liegen, werden auf Null gesetzt. Der Schwellwert wird zunächst auf einen unendlichen Wert initialisiert und am Ende auf die Durchschnittsleistung des Signals unterhalb des Schwellwertes gesetzt. Das teilweise auf Null gesetzte Signal im Frequenzbereich wird anschließend durch eine IFFT wieder in den Zeitbereich zurück transformiert. Ohne eine Erhöhung der Komplexität kann mit dieser Methode auch mehr als eine Störung erkannt und ausgelöscht werden. Die genaue Erkennung der Störsignale ist möglich und konzentriert sich auf schmalbandige Pins [27].

Da bei der FFT sogenannte Leck-Effekte auftreten können, wird vor diese eine Fensterfunktion gesetzt, um diese zu mindern. Bei Verwendung einer Fensterfunktion nimmt die spektrale Auflösung im Frequenzbereich zu. Im vorhandenen Algorithmus kann bei der Fenstermethode zwischen einem rechteckigen Fenster, der Hann-Funktion und der Blackman-Harris gewählt werden. Diese sind im Vergleich in Abbildung 26 dargestellt.

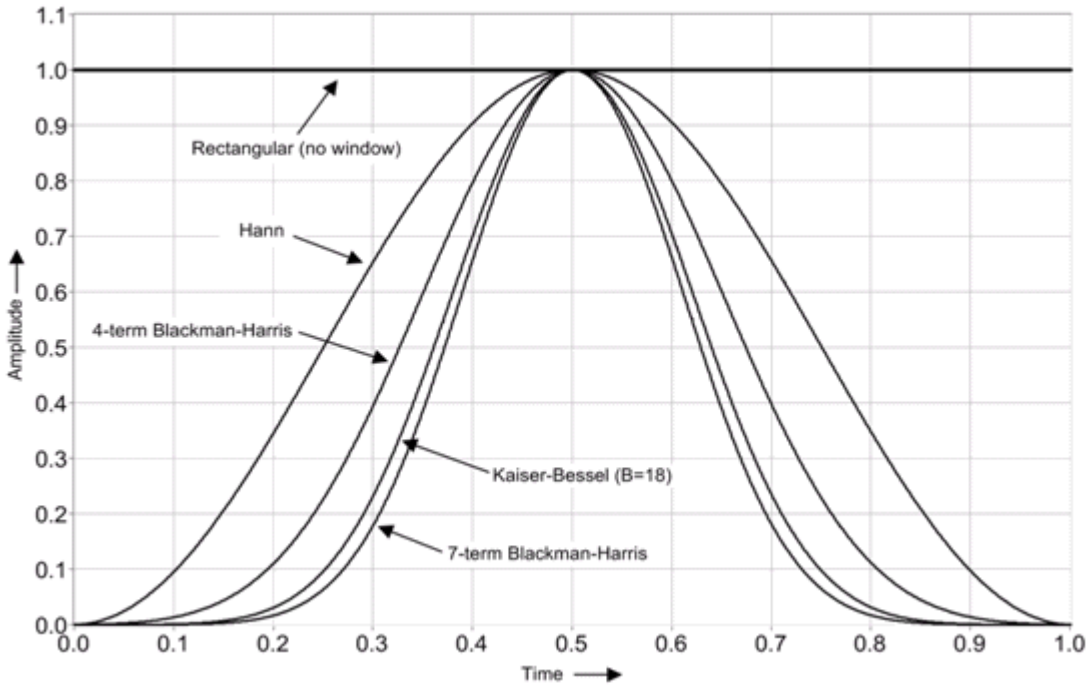


Abbildung 26: Zeitbereich der Fenstermethoden (Quelle: [30])

Die Rechteckfunktion bewirkt keine Fensterung des Eingangssignals vor der FFT. Die Hann-Fensterfunktion ist für Rauschen geeignet, was eine bessere Frequenzauflösung als andere Fensterfunktionen bietet. Dabei stellen moderate Nebenkeulen kein Problem dar. Die Blackman-Harris Fensterfunktion ist ein gutes Allzweckfenster mit Seitenkeulendämpfung in den hohen 90er dB und besitzt eine mäßig breite Hauptkeule [30].

Einen weiteren Parameter für den FFT-Algorithmus stellt die Überlappung (engl.: Overlapping; Abk.: OV) dar. Einstellbar ist eine Überlappung von 0, 0,5 und 0,75. In Abbildung 27 ist die Funktionsweise der Überlappung grafisch dargestellt. Die Überlappung bezieht sich in diesem Algorithmus auf den gesamten Pfad, von der Fensterfunktion bis zur inversen FFT.

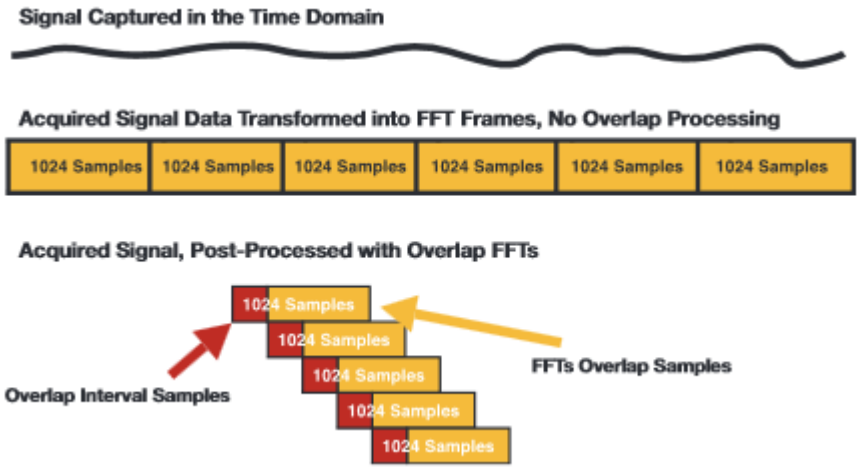


Abbildung 27: Beispiel der Überlappung (Quelle: [31])

Bei 0 erfolgt keine Überlappung, mit 0,5 erfolgt eine Überlappung des halben Eingangssignals und bei 0,75 erfolgt eine Überlappung von 75% der Samples. Dabei wird die Eingangsfolge in Teilfolgen zerlegt. Wird die Überlappung auf 50% gesetzt, resultiert ein doppelter Rechenaufwand für diesen Algorithmus. Mit der Anwendung einer Überlappung erhöht sich die Zeitauflösung und die Frequenzauflösung bleibt identisch. Das Verhalten der FFT mit Überlappung verhält sich analog zur STFT.

5.4.2 Karhunen-Loève-Transformation

Bei der Karhunen-Loève-Transformation (KLT) erfolgt eine Zerlegung der Eingangssignale in ihre Eigenvektoren und Kovarianz-Matrizen. Diese Art der Transformation benötigt einen hohen Rechenaufwand. Vergleicht man die KLT mit der FFT bietet diese zusätzlich eine stochastische Beschreibung des Signals und unbegrenzte Grunddefinitionen. Somit ist die KLT eine individuellere Signalzerlegung als die FFT [27].

5.4.3 Wavelet-Transformation

Die Wavelet-Transformation (WT) ist ein Verfahren zur Zeit-Frequenz-Transformation. Neben der bekannten Anwendung zur Bildkomprimierungstechnik kann die WT auch für das Unterdrücken von Störsignalen angewendet werden. Die Technik der WT ist in der Lage, eine Menge von Störungen zu erkennen. Als unvermeidbare Konsequenz können von den genutzten QMFs (Quadrature Mirror Filter) entlang der Baumzerlegung Artefakte auftreten. Nachdem ein Artefakt generiert wurde, können diese als Störkomponenten entlang der Baumstruktur bis hin zur Kombinierung von realen Störkomponente oder anderen Artefakten weitergegeben werden. Die Erzeugung von Artefakten ist bei den meisten Störern möglich und kann bei mehreren auswählbaren QMFs unterdrückt werden. Die Zerlegung während der Wavelet-Analyse wird in zwei Schritten durchgeführt. Im ersten Schritt wird der komplette Baum in drei Ebenen Ls aufgeteilt. Am Ende wird die Leistung des GNSS-Signals abgeschätzt, wobei der relative Schwellwert bestimmt wird. Die Anzahl der Ebenen Ls kann durch den Benutzer definiert werden. Die verschiedenen Frequenzanteile werden weitergeben, wenn der Schwellwert überschritten wird. Daraus folgt die Auslöschung der Störkomponenten und die Wiederherstellung des Signals ohne Störkomponente. Um mit der Wavelet-Synthese das Signal zu rekonstruieren, werden nur Blätter ohne Störung verwendet [27] [47].

5.4.4 Pulse Blanking

Der Pulse Blanking Algorithmus mit Leistungserkennung eignet sich für die Unterdrückung von gepulsten Signalen. Dieser Algorithmus arbeitet im Zeitbereich, was keinen intensiven Rechenaufwand für die Transformation in den Frequenzbereich benötigt. Überschreitet die Amplitude im Zeitbereich einen gewissen Schwellwert, erfolgt eine Nullsetzung aller Werte. Durch das Nullsetzen der Störung erfolgt ebenso eine Reduzierung der GNSS-Signalenergie, da diese ebenfalls in diesem Bereich enthalten ist [27].

5.4.5 Notch Filter

Der Notch Filter, auf Deutsch „Kerbfilter“ genannt, ist ein Filter zum Herausfiltern von Frequenzen innerhalb eines engen Frequenzbereichs. Es handelt sich um einen schmalbandigen Typ, idealerweise zum Auslöschen einer Frequenz. Der Notch Filter Erkennungsmechanismus basiert auf dem Erkennen des Null Filter Modulo. Wenn der Modulo über dem Schwellwert (basierend auf dem eingestellten false alarm probability) ist, greift der Notch Filter. Ist das nicht der Fall, werden die Eingangssignale nicht prozessiert. Der Notch Filter ist in der Lage, kontinuierliche Schwingungen und schmalbandige Störungen in einer effektiven Art und Weise zu erkennen und zu unterdrücken. Ist eine breitbandige Störung vorhanden (aber immer noch gleich als die gesamte GNSS-Bandbreite), so ist dieser in der Lage, die Störung zu erkennen, aber nicht effektiv zu unterdrücken. Der Notch Filter ist nicht in der Lage, gepulste Störungen zu erkennen und zu unterdrücken [27].

5.5 Auswertung der Algorithmen

In diesem Kapitel wurden die zur Verfügung stehenden Algorithmen, welche in Tabelle 10 aufgelistet sind, zur Störsignalunterdrückung auf die vier ausgewählten und aufgezeichneten störbehafteten GNSS-Signale in Tabelle 7 angewendet. Die Auswertungen hinsichtlich der Unterdrückung von Störsignalen erfolgte mit dem in Kapitel 5.2, Kapitel 4.4 und Kapitel 4.2.1.1 vorgestellten Tools. Die Ergebnisse basieren auf dem Galileo E1 Band mit der Satellitennummer 23.

Zunächst wurde bei allen zu analysierenden Szenarien die FFT angewendet. Am Anfang erfolgte eine visuelle Überprüfung im TDK, ob überhaupt ein Unterdrückungserfolg vorhanden ist. Da bei dem FFT-Algorithmus viele verschiedene Eingangsparameter wählbar sind, wurden systematisch immer nur bei einem Parameter alle verschiedenen Möglichkeiten getestet. Der erste betrachtete Parameter der FFT war die Länge. Weitere Parameter wurden mit den zuvor als empfohlenen Eigenschaften der jeweiligen Entwickler der Algorithmen festgelegt. Der Überlappungsfaktor wurde mit 0,5, als Fensterfunktion Hann und der Schwellwert bei 4,0 festgelegt. Wurde die beste FFT-Länge bestimmt, erfolgte das Ermitteln der geeigneten zusätzlichen Parameter. Nach der Bestimmung der besten Parameter für die Unterdrückung mit der FFT, wurden noch der KLT, WT, Pulse Blanking und der Notch Filter mit den einstellbaren Parametern auf den Erfolg der Störsignalunterdrückung untersucht. Diese wurden ebenfalls zuerst visuell auf einen möglichen Unterdrückungserfolg untersucht, bevor das digitalisierte GNSS-Signal mit dem TDK durchgelaufen wurde.

Es hat sich gezeigt, dass im Schnitt für die Auswertung eines digitalisierten GNSS-Signals mit der FFT, WT und Notch Filter eine CPU-Auslastung von ca. 10% (verwendeter CPU-Typ: Intel® Core™ i7 X980 @ 3,33 GHz) eine langen Durchlaufzeit benötigt. Eine genauere Betrachtung über die Qualität der Echtzeitfähigkeit erfolgt in Kapitel 5.6. Um die freien, nicht verwendeten Ressourcen der CPU nutzbar zu machen, wurden Duplikate des TDK erstellt. Somit war es gleichzeitig möglich vier verschiedene Algorithmen zur Störsignalunterdrückung mit vier parallel laufenden TDKs zu prozessieren. Da die KLT eine

sehr hohe Rechenleistung benötigt, war es nicht möglich bei diesem mehrere parallel zu betreiben. Ebenfalls wurde der Pulse Blanking Algorithmus nur in einem geöffneten TDK prozessiert, da dieser Algorithmus in Echtzeit durchläuft und die Samples innerhalb weniger Minuten prozessiert wurden.

Sind die Samples durch das TDK durchgelaufen und vollständig prozessiert, wurden noch die jeweiligen RINEX-Dateien der neu entstandenen Samples erzeugt. Dazu wurden die Samples mit dem Softwareempfänger im Post-Processing eingelesen und prozessiert. Daraus erfolgte wieder eine RINEX-Datei, welche mit der RINEX-Datei der Samples ohne Unterdrückung verglichen wurde. Die RINEX-Dateien wurden mit dem RINEX-Viewer ausgewertet, welcher die grafischen Plots erzeugte.

Bei den anschließenden Auswertungen der jeweiligen Szenarien wurden die besten Parameter der jeweiligen Algorithmen den RINEX-Dateien gegenüber gestellt, bei welchen keine Unterdrückung erfolgte.

Die erzeugten Plots zeigen das C/N_0 des GNSS-Signals in Abhängigkeit der Leistung der Störung an. Ein hoher Wert des C/N_0 in der vertikalen Achse spricht für eine gute Signalqualität, welche mit steigender Störleistung in der horizontalen Achse abnimmt.

5.5.1 Vier kontinuierliche Schwingungen (Szenario #1)

Für die Vorauswahl der Algorithmen wurde die FFT, die WT und der Notch Filter bei einer Störung mit vier kontinuierlichen Schwingungen gewählt. Da die KLT ein komplexer Algorithmus für komplexe Störsignale ist und dieses Szenario mit den ausgewählten Algorithmen gut unterdrückbar ist, wurde dieser in diesem Störsignal nicht behandelt. Der Pulse Blanking wurde ebenfalls in diesem Szenario nicht behandelt, da dieser wie in Kapitel 5.4.4 beschrieben nur für gepulste Störsignale von Bedeutung ist. In Abbildung 28 erfolgt die Übersicht aller angewendeten Algorithmen mit den besten Parametern auf das Szenario #1 mit vier kontinuierlichen Schwingungen als Störsignal in einem Plot.

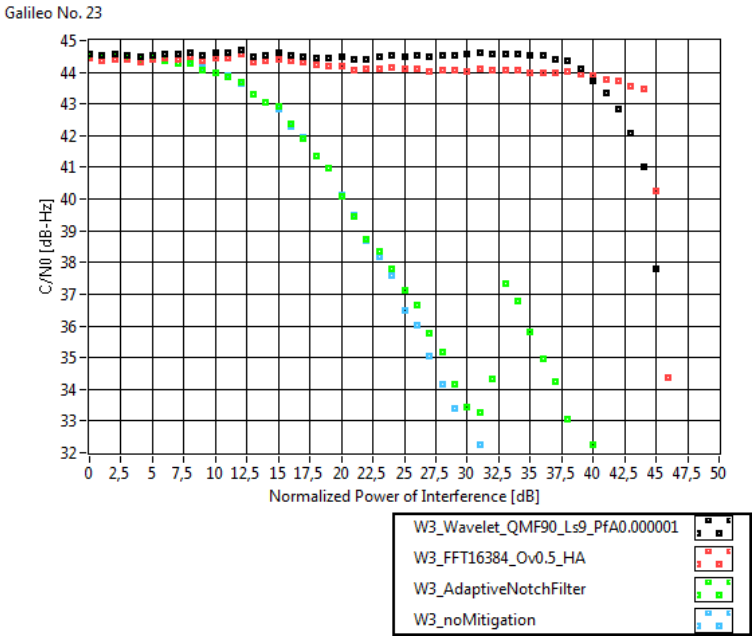


Abbildung 28: Ergebnis der Unterdrückung von vier kontinuierlichen Schwingungen

Das beste Ergebnis bei der Unterdrückung von vier kontinuierlichen Schwingungen hat sich mit einer hohen Länge der FFT erwiesen. Hierbei ist bis zu einer Störleistung von 40 dB eine Degradierung des C/N_0 um nur 0,5 dB-Hz vorhanden. Dies spricht für einen sehr guten Unterdrückungserfolg. Die WT erfüllte ebenfalls die Anforderungen zum Unterdrücken dieser Art von Störsignal. Lediglich bei einer Leistung von 37,5 dB beginnt auch das C/N_0 stark zu sinken. Der Notch Filter ist in seiner Eigenschaft für die Unterdrückung bei diesem Szenario nicht funktionsfähig. Sobald die Störkomponente wirkt, sinkt auch das C/N_0 trotz einer Erhöhung des Wertes für die Anzahl der nScans auf die Anzahl der enthaltenen kontinuierlichen Schwingungen. Auffällig ist, dass bei einer Leistung der Störung von 32,5 dB kurzzeitig das C/N_0 steigt. Bei nur einer kontinuierlichen Schwingung als Störkomponente arbeitet der Notch Filter einwandfrei.

5.5.2 Gefiltertes AWGN B=100 kHz (Szenario #2)

Für das gefilterte Rauschen als Störsignal wurden die FFT, KLT und WT gewählt und mit verschiedenen Parametern getestet. Der Notch Filter und das Pulse Blanking sind zum Unterdrücken eines Rauschens als Störung nicht geeignet. Der Notch Filter ist nur in der Lage, kontinuierliche Schwingungen und schmalbandige Störungen zu erkennen und zu unterdrücken. Der Pulse Blanking ist nur für gepulste Signale geeignet. In Abbildung 29 erfolgt die Übersicht aller angewendeten Algorithmen auf das Szenario #2 mit einem gefilterten Rauschen der Bandbreite von 100 kHz als Störsignal in einem Plot.

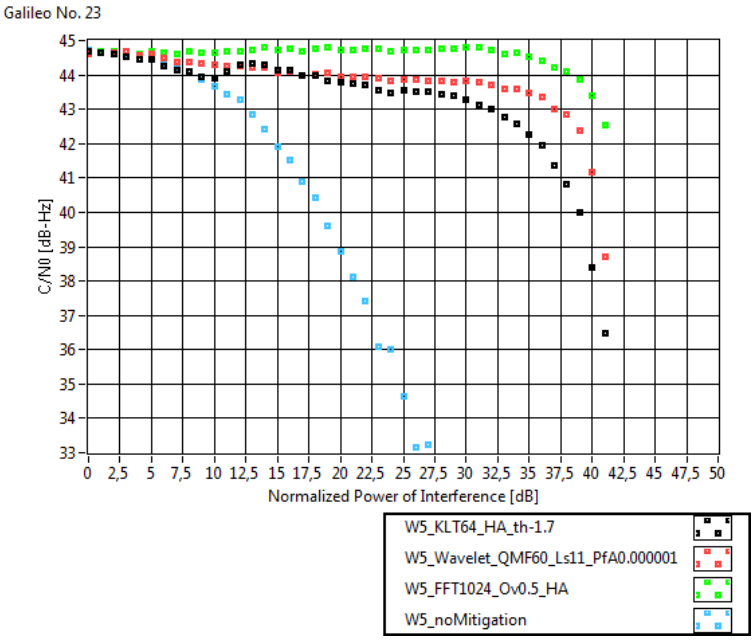


Abbildung 29: Ergebnis der Unterdrückung von gefiltertes Rauschen

Der FFT-Algorithmus erwies sich mit einer Länge von 1024 als effektivster Algorithmus um gefiltertes Rauschen zu unterdrücken. Ab einer Störleistung von 32,5 dB beginnt das C/N_0 zu sinken und fällt rapide ab. Dieses Verhalten ist auch bei den anderen angewendeten Algorithmen zu sehen. Ebenso ist die WT geeignet, jedoch mit dem Nachteil, dass kurz nach Eintreten der Störung das C/N_0 um 1 dB sinkt bis es auf eine Leistung von 35 dB abfällt. Die KLT ist ebenfalls in der Lage die Störung zu unterdrücken, allerdings ist ab einer Leistung von 25 dB das C/N_0 um 0,5 dB-Hz geringer im Vergleich zur WT. Erhöht sich die Bandbreite des Rauschens, wird es umso schwieriger, dieses zu bereinigen. Bei einer Bandbreite von 2 MHz arbeitet kein Algorithmus wirkungsvoll.

Der FFT-Algorithmus lieferte das höchste C/N_0. Um eine eindeutige Aussage über diese Erkenntnis zu treffen, sind weitere Untersuchungen der Transformationsarten und der Implementierung bei der Störsignalunterdrückung notwendig. Jedoch wird ein möglicher Grund gegenüber der WT erläutert: Mit steigender Frequenz des Störsignals sinkt bei der WT die Frequenzauflösung und die Zeitauflösung erhöht sich. Steigt dementsprechend die Frequenz des Störsignals im GNSS-Spektrum, bietet die WT eine signifikant schlechtere Frequenzauflösung. Somit geht bereits bei einer schmalbandigen Störung mit mittlerer und höherer Frequenz deutlich mehr Energie des GNSS-Signals als bei der FFT verloren, was zu einer minderen C/N_0 führt.

5.5.3 Chirp-Signal (Szenario #3)

Das Chirp-Signal stellt das simulierte Störsignal dar, welches von Jammern ausgesendet wird. Dieses Störsignal wurde für die Unterdrückung mit der FFT, KLT und WT angewendet. Das Pulse Blanking und der Notch Filter sind bei dieser Störung ungeeignet. In Abbildung 30 erfolgt die Übersicht aller angewendeten Algorithmen auf das Chirp-Signal.

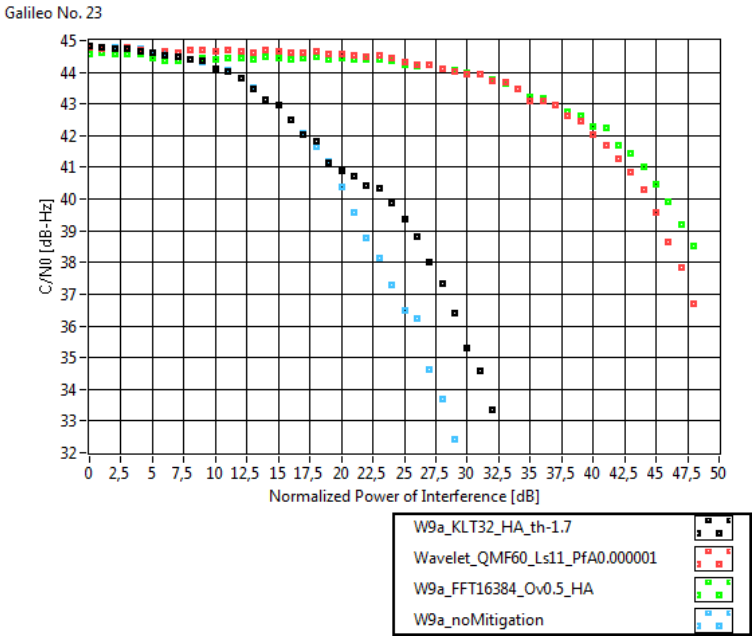


Abbildung 30: Ergebnis der Unterdrückung von einem Chirp-Signal

Der Unterdrückungserfolg ist bei der FFT und WT nahezu identisch. Bei Beiden beginnt eine Degradierung des C/N_0 ab 25 dB. Bei einer hohen Leistung der Störung weist die FFT eine geringfügig bessere Unterdrückung als die WT auf. Der KLT greift zu Beginn nicht. Erst ab einer Leistung von 20 dB beginnt dieser minimal zu wirken. Für die Länge der FFT ist ein hoher Wert geeignet.

5.5.4 Radar (Szenario #4)

Für die Unterdrückung eines gepulsten Chirp-Signals, wie es beim Radar vorkommt, wurde die FFT, KLT, WT und der Pulse Blanking Algorithmus mit verschiedenen Parametern verwendet. Der Notch Filter kommt nicht zum Einsatz. In Abbildung 31 ist die C/N_0 Degradierung in Abhängigkeit der Leistung des Störsignals auf die verschiedenen Algorithmen zu sehen.

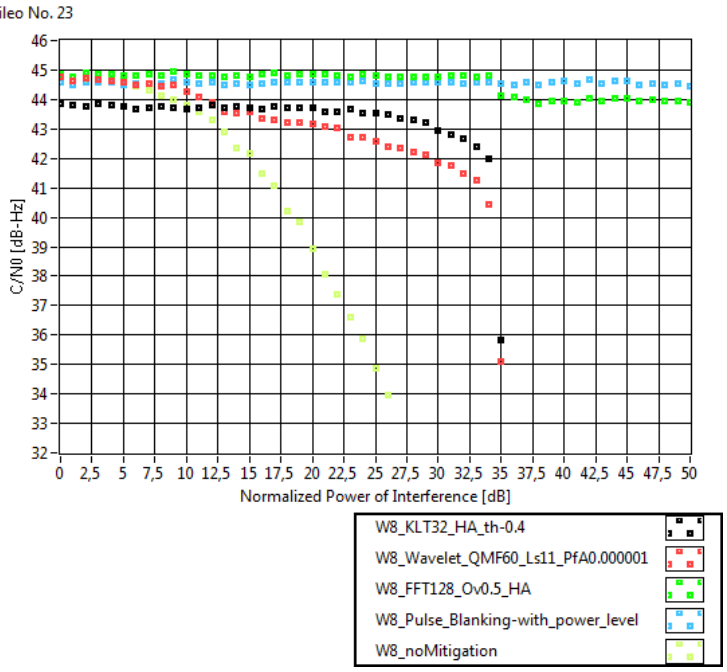


Abbildung 31: Ergebnis der Unterdrückung von einem Radar

Der Pulse Blanking sowie die FFT bieten eine sehr gute Möglichkeit, um ein gepulstes Störsignal zu unterdrücken. Der Pulse Blanking weist einen durchgehend konstanten Wert des C/N_0 bei stärker werdender Leistung des gepulsten Störsignals auf. Der konstante Wert des C/N_0 ist darauf zurückzuführen, dass die Störung nur in einem bestimmten Intervall auftritt und dieser im Zeitbereich entfernt wird. Es spielt keine Rolle, welche Leistung die Störung besitzt. Würde der gepulste Intervall erhöht werden, würde dies eine Degradierung des C/N_0 nach sich ziehen. Bei der FFT hat sich eine kleine Länge bewährt. Bis hin zu einer Leistung von 34 dB des Störsignals arbeitet dieser Algorithmus minimal besser als der Pulse Blanking. Danach findet sprunghaft eine Degradierung von C/N_0 um 1 dB statt. Nach dieser Minderung geht der C/N_0 kontinuierlich weiter. Die WT weist bei einer Störung eine lineare Abschwächung des C/N_0 bei kontinuierlicher Steigerung der Störleistung auf. Bei einer Leistung von 34 dB fällt der C/N_0 bis hin zum kompletten Signalverlust ab 35 dB stark ab. Die KLT verhält sich ähnlich wie die WT. Ist noch keine Störung vorhanden, ist direkt eine Degradierung des C/N_0 um 1 dB sichtbar. Auffällig ist, dass ab einer Störleistung von 35 dB alle Algorithmen außer dem Puls Blanking einen Qualitätsverlust aufweisen.

5.6 Qualität der Echtzeitfähigkeit

In dem letzten Kapitel zum Unterdrücken von Störsignalen wurden diese in Bezug auf die Qualität der Echtzeitfähigkeit untersucht. CPU-gestützte Algorithmen zur Störsignalunterdrückung benötigen heute eine sehr hohe Rechenleistung auf High-End PCs. Heutige Chips eines Smartphones verfügen über eine sehr geringe Leistungsaufnahme, was auf mobilen Empfangsgeräten von großer Bedeutung ist, da diese mit einem Akku betrieben werden und eine möglichst lange Laufzeit erreichen sollen.

Der Begriff „Echtzeit“ bezieht sich in der Informationstechnik auf das rechtzeitige Eintreten eines Ereignisses innerhalb einer vorbestimmten Zeitspanne. Wenn hier von Echtzeit die

Rede ist, ist gemeint, dass die erforderliche Menge an Daten in einem definierten Zeitfenster bearbeitet werden muss.

Um eine Aussage über die Echtzeitfähigkeit eines Algorithmus zu machen, muss zuerst definiert werden, welche Zeit es bei der Bearbeitung einzuhalten gilt. Die Abtastrate bei der Digitalisierung der GNSS-Signale ist hardwarebedingt durch das NavPort-4 Front-End für den Softwareempfänger mit 20,48 MHz vorgegeben. Die Paketgröße ist ebenfalls durch den Softwareempfänger mit 675.840 festgelegt. Durch die Formel

$$tmax = \frac{Paketgröße}{Abtastrate}$$

ergibt sich eine maximal erlaubte Zeit von 33 ms, welche nicht überschritten werden darf, um als echtzeitfähig zu gelten. Alle über dieser Zeit liegenden Algorithmen gelten als nicht echtzeitfähig. Im TDK ist im Reiter „Deterministic“ ein Indikator, welcher die Durchschnittszeit pro Schleifendurchgang für das Einlesen und das Ausgeben der Samples anzeigt. Das Erkennen von Störsignalen ist in der Regel immer echtzeitfähig. Um eine Störung zu erkennen, reicht es aus, nur eine Sequenz in einem definierten Intervall der Samples zu untersuchen. Tabelle 11 stellt die Ergebnisse der Echtzeitanalyse der verwendeten Algorithmen dar.

Tabelle 11: Übersicht der Echtzeitfähigkeit der Algorithmen

Typ mit Parameter des Algorithmus	CPU [%]	Zeit [ms]	Pass
FFT1024_Ov0.5_HA	6-10	470	X
FFT16384_Ov0.5_HA	6-10	140	X
KLT32_HA_th-1.7	32	1450	X
WT	8-10	430	X
Pulse Blanking	1-2	2,3	OK
Notch Filter	8-10	440	X

Es ist erkenntlich, dass nur der Pulse Blanking die Echtzeitfähigkeit über den notwendigen Wert von 33 ms erfüllt und somit als voll echtzeitfähig für die Unterdrückung von Störsignalen im GNSS-Bereich gilt. Zusätzlich wurde noch die Auslastung der CPU in die Tabelle aufgenommen. Der Pulse Blanking zeigt bei sehr schneller Verarbeitung eine geringe CPU-Auslastung.

Da sich die FFT im vorherigen Kapitel als der effizienteste Algorithmus in Bezug auf die Abdeckung aller vorkommenden Arten von Störungen herausgestellt hat, erfolgt bei der FFT eine genauere Auswertung zur Qualität der Echtzeitfähigkeit. Die FFT wurde mit verschiedenen Parameter, welche die FFT-Länge, Fensterfunktion und Überlappungsfaktor darstellen verglichen. Mit den letzten zwei Parametern wurden drei verschiedene Einstellungen gewählt, welche alle mit einer FFT-Länge von 2^7 bis 2^{16} durchlaufen wurden und in einem Diagramm dargestellt sind. Das aus den Ergebnissen resultierende Diagramm ist in **Fehler! Verweisquelle konnte nicht gefunden werden.** dargestellt.

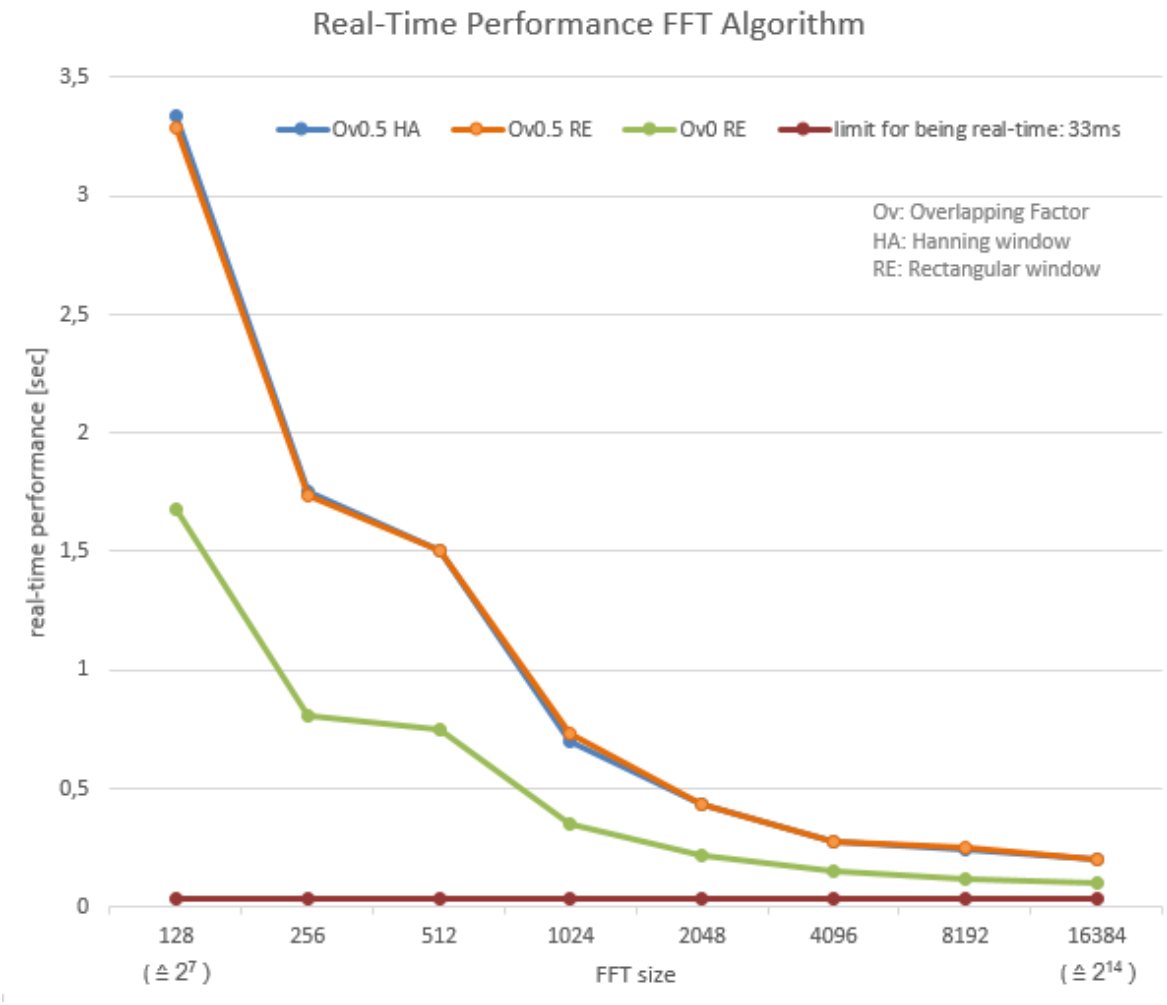


Abbildung 32: Echtzeitfähigkeit des FFT-Algorithmus

Es ist erkennbar, dass der FFT-Algorithmus in keiner Weise die Anforderungen von 33 ms für die Echtzeitfähigkeit erfüllt. Anhand einer Fachrecherche ist es möglich, diesen Algorithmus in Echtzeit zu betreiben [32]. Eine FFT mit der Länge von 8192 kann innerhalb von 200 µs berechnet werden. Im TDK wird in Bezug auf den Softwareempfänger mit einer Paketgröße von 675.840 gearbeitet:

$$tmax = \frac{675840}{8192} * 2 * 200\ \mu s = 33\ ms$$

Mit obiger Formel wird die Echtzeit von 33 ms eingehalten. Zusätzlich lassen sich mit geeigneter Parallelisierung noch weitaus bessere Werte realisieren. Der Faktor 2 in obiger Formel bezieht sich auf die sequentielle Abarbeitung der CPU. Wenn die FFT und IFFT parallelisiert werden, wird der Algorithmus in der Hälfte der Zeit mit 16,5 ms abgearbeitet.

Zwar lässt sich mit anderen Programmiermethoden des FFT-Algorithmus durchaus die Einhaltung der Echtzeitfähigkeit garantieren, aber nur mit großer Rechenleistung.

5.7 Ergebnis

Anhand der Ergebnisse der Auswertung für das Unterdrücken von Störsignalen im GNSS-Bereich erfolgt eine Abschätzung über die Verwendbarkeit und Rentabilität der in Betracht gezogenen Algorithmen.

Die Unterdrückung mittels der FFT bietet eine solide Methode, um alle Arten von Störquellen mit einem meist sehr guten Ergebnis abzudecken. Nachteilig hat sich herausgestellt, dass durch den großen Rechenaufwand der FFT diese meist noch nicht auf einem CPU-gestützten System echtzeitfähig ist. Um eine Echtzeitfähigkeit mit dieser Methode zu erreichen, ist es notwendig, auf alternative Möglichkeiten für das Prozessieren für die digitale Signalverarbeitung umzusteigen. Dies könnte durch einen FPGA erfolgen.

Die KLT befindet sich noch am Anfang seiner Entwicklungsphase für die Störsignalunterdrückung. Dieser bietet durchaus mehr Potenzial in Bezug auf den Unterdrückungserfolg, die erforderliche Rechenleistung und der daraus resultierenden Echtzeitfähigkeit.

Die WT liefert ein zufriedenstellendes Ergebnis, jedoch wurde festgestellt, dass der Unterdrückungserfolg bei diesem Algorithmus in jedem betrachteten Szenario weniger Erfolg lieferte als die FFT. Theoretisch müsste die WT aufgrund der Tatsache der höheren Komplexität mehr Erfolg zeigen. Anscheinend ist dieser in seinem Umfang bei der Implementierung aufgrund der hohen Komplexität nicht ausreichend funktionsfähig.

Der Pulse Blanking ist für gepulste Störsignale sehr gut geeignet. Da bei diesem Algorithmus keine Transformation in den Frequenzbereich stattfindet, ist dieser bei der Unterdrückung voll echtzeitfähig. Bei nicht gepulsten Signalen ist dieser nicht in der Lage, das Störsignal erfolgreich zu unterdrücken und somit ungeeignet.

Der Notch Filter ist durch seine minimale Anzahl an sukzessiven nScans von maximal vier nur für kontinuierlich Schwingungen beschränkt geeignet. Jedoch ist dieser schon bei einer Anzahl von vier kontinuierlich enthaltenen Störungen nicht mehr in der Lage diese zu unterdrücken.

Als bester Algorithmus für alle Szenarien für die Störsignalunterdrückung in Bezug auf die GNSS-Signalqualität mit dem Indikator des Träger-zu-Rauschverhältnisses (C/N_0) hat sich die FFT erwiesen. Mit den in Tabelle 12 aufgelisteten Parametern hat sich der durchschnittlich auf alle angewendeten Szenarien beste Unterdrückungserfolg herausgestellt. Nun gilt es, diesen FFT-Algorithmus auf einem FPGA zu implementieren.

Tabelle 12: Ideale Parameter des FFT-Unterdrückungsalgorithmus

Eigenschaften der FFT	**Parameter**
Länge	1024
Fensterfunktion	Hann
Überlappungsfaktor	0,5
Schwellwert	4,0

6 GNSS-Störsignalunterdrückungseinheit

Die CPU-gestützten Algorithmen wurden alle für das Prozessieren auf einem Rechner entwickelt. Es hat sich gezeigt, dass hinsichtlich der Qualität auf Echtzeitfähigkeit die Rechenleistungen von Prozessoren oft nicht ausreichend sind. Für das Umsetzen von rechenintensiven Algorithmen eignet sich ein FGPA durch seine parallele Verarbeitung besonders gut. Es soll damit eine GNSS-Störsignalunterdrückungseinheit auf einem SDR-System realisiert werden. Auf dem FPGA soll die Möglichkeit bestehen, einen gewählten Algorithmus zur Störsignalunterdrückung zu implementieren. Das SDR-System soll als GNSS-Front-End oder als GNSS-Repeater mit Möglichkeit zur Störsignalunterdrückung dienen. Die Störsignalunterdrückungsfunktion soll zu- und abschaltbar sein. Somit wird ein flexibles Gerät zum Entwickeln von Algorithmen zur Störsignalunterdrückung auf einem FPGA bereitgestellt. Da ein SDR-System von National Instruments die Samples im Basisband als I/Q-Daten bereitstellt, aber der zu empfangende Softwareempfänger Samples im Zwischenfrequenzbereich (ZF) benötigt, gilt es, diese Transformation ebenfalls auf dem FPGA umzusetzen.

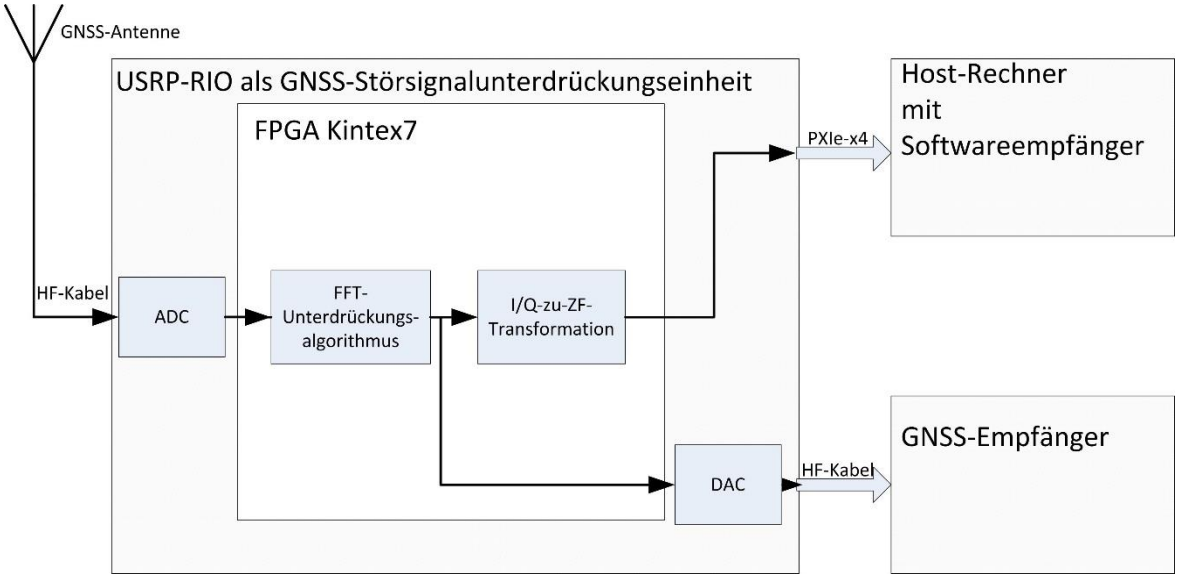


Abbildung 33: Ablaufplan der GNSS-Störsignalunterdrückungseinheit

Bei der Auswertung der zur Verfügung stehenden Algorithmen auf störbehaftete GNSS-Signale wurde in Kapitel 5 festgestellt, dass der FFT-Unterdrückungsalgorithmus der Effektivste in Bezug auf den Unterdrückungserfolg von verschiedenen vorkommenden Störquellen ist. Dieser deckt mit den verwendeten Parametern ein weites Spektrum vorkommender Störsignale ab. Es war auch ersichtlich, dass dieser bei der Implementierung auf einem Rechner mit CPU-gestützten Algorithmen für das Erreichen der Echtzeitfähigkeit eine sehr große Rechenleistung benötigt. Nur durch gezieltes Multithreading war es möglich, diesen unter voller Rechnerlast in Echtzeit zu betreiben. Deshalb wird dieser Algorithmus mit seinen besten ermittelten Parametern auf dem FPGA des SDR-Systems implementiert.

6.1 NI USRP-2952R

Das NI USRP-2952R von National Instruments, dargestellt in Abbildung 34, ist ein SDR-System mit einem Hochfrequenz-Transceiver, welches mit einem XILINX-7 K410T FPGA zur freien Programmierung mittels LabVIEW ausgestattet ist [33]. Mit dem Begriff „Software Defined Radio“ beschreibt man das Bestreben, möglichst die gesamte Signalverarbeitung eines Hochfrequenzsenders oder –empfängers mithilfe anpassbarer Software in Hardware abzubilden [34]. Solche SDR-Systeme gibt es für verschiedene Frequenzbereiche. Für die Aufzeichnung von Navigationssatellitensignalen ist darauf zu achten, dass das verwendete SDR-System den Frequenzbereich von GNSS abdeckt. Dabei gilt es, das analoge GNSS-Signal zu digitalisieren, welches als Sample auf dem Rechner gespeichert oder in Echtzeit weiterverarbeitet wird.

Das NI USRP-2952R von National Instruments bietet alle Voraussetzungen für die Verwendung als GNSS-Front-End, GNSS-Repeater und um die Algorithmen zur digitalen Signalverarbeitung für die Störsignalunterdrückung direkt auf dem integrierten FPGA zu implementieren. Die gewünschte Signalverarbeitung des empfangenen Signals wird direkt nach dem Empfang auf dem FPGA vorgenommen. Somit wird die fest definierte Signalverarbeitung direkt im Front-End oder Repeater erledigt. Wichtige Parameter des verwendeten SDR-Systems sind in Tabelle 13 aufgelistet. Die Anbindung an den Host-Rechner erfolgt über einen PCIe x4 Bus.

Tabelle 13: Spezifikationen des USRP-2952R (Quelle: [33])

Eigenschaften	Parameter
Frequenzbereich	400 MHz – 4,4 GHz
Bandbreite	40 MHz
Frequenzgenauigkeit	25 ppb (unlocked)
ADC	120 MS/s mit 14 Bit
DAC	400 MS/s mit 16 Bit

Abbildung 34: NI USRP-2952R (Quelle: [33])

Für die Erzeugung der internen Schwingung ist ein beheizter Quarzoszillator (engl.: Oven Controlled Crystal Oscillator; Abk.: OCXO) verbaut. Für ein erfolgreiches Tracken der Satellitenposition muss dieser über eine ausreichende Qualität verfügen, um Langzeitstabilität zu gewährleisten. Alternativ kann eine externe Quelle verwendet werden, um eine höhere Genauigkeit zu erreichen [33].

Bei dem USRP-RIO ist bereits ein Beispielprogramm enthalten, mit dem Signale empfangen und gesendet werden können. In diesem FPGA-Code gilt es, beide Algorithmen (I/Q-zu-ZF-Transformation und FFT-Unterdrückungsalgorithmus) zu implementieren. Zusätzlich muss die Steuersoftware den Bedürfnissen angepasst werden.

6.2 Xilinx Kintex-7 K410T

Die entscheidende Rolle für das Implementieren der vorhandenen Algorithmen auf einem SDR-System stellt der darauf verbaute FGPA dar. Ein FPGA ist ein wiederprogrammierbarer Schaltkreis, bestehend aus Logikzellen und programmierbaren Verbindungsleitungen. Die primären Vorteile eines FPGAs liegen in seiner Parallelität, Flexibilität und Konfigurierbarkeit. Mit der Parallelität ist dieser sehr gut für die Implementierung von rechenintensiven Algorithmen geeignet, welche bei der Störsignalunterdrückung zum Einsatz kommen. Da es sich bei der Implementierung eines Algorithmus um eine Prototypenrealisierung handelt, ist die Konfigurierbarkeit und Flexibilität von Bedeutung, um diesen stetig an erweiterte Anforderungen anpassen und ergänzen zu können. Die Nachteile liegen im Vergleich zur Verwendung einer anwendungsspezifischen integrierten Schaltung (engl.: application-specific integrated circuit; Abk.: ASIC) in der geringeren Logikdichte, wodurch höhere Verzögerungszeiten auftreten. Die Leistungsaufnahme eines FPGAs ist ebenfalls höher und die Anschaffungskosten sind teuer. Um den Aufbau und die Funktionsweise eines FPGAs zu erläutern, werden anhand des im USRP-RIO verbauten Xilinx FPGA Kintex-7 XC7K410T die Hauptelemente betrachtet. Die wichtigsten Eigenschaften eines FPGA sind in Tabelle 14 mit den Spezifikationen des verwendeten FPGAs aufgelistet.

Tabelle 14: Spezifikationen des Xilinx Kintex-7 XC7K410T FPGA [35]

Elemente	**Anzahl**
Logikzellen	406720
Konfigurierbare Logikblöcke (CLBs)	63550
max. verteilter RAM	5663 kBit
DSP-Slices	1540
Block RAM	28620 kBit
Max. User I/O	500

Der Konfigurationsspeicher des FPGAs bestimmt die Funktionalität der Logikzellen und Verbindungen. Die Verbindungen werden bei FPGAs von Xilinx durch die programmierbare Verdrahtung auf SRAM-Basis gelöst. Der Vorteil gegenüber der Antifuse Methode, die das durchbrennen der hochohmigen Verbindung bewirkt, besteht darin, dass mit der SRAM-basierten Methode die Verbindungen beliebig oft neu konfigurierbar sind [36]. Als Logikzellen stehen 63.550 konfigurierbare Logikblöcke (engl.: Configurable Logic Blocks; Abk.: CLBs) mit jeweils vier Lookup-Tabellen (engl.: Look Up Tables, Abk.: LUTs) und acht Flip Flops zur Verfügung. Somit enthält der FPGA insgesamt 254.200 LUTs und 508.400 Flip Flops. Bestimmte Operationen benötigen mit der Realisierung von LUTs und Flip Flops

eine große Anzahl dieser Ressourcen, weshalb sogenannte DSP-Slices zur Verfügung stehen. Diese enthalten speziell vorkonfigurierte Schaltungen für die Umsetzung bestimmter Aufgaben. Jeder DSP-slice beinhaltet einen Voraddierer, 25x18 Bit Multiplizier, Addierer und einen Akkumulator. Neben dem verteilten RAM, welcher mit den Logikblöcken generiert wird, ist zusätzlich noch ein Block RAM auf dem FPGA vorhanden [35]. Der Vorteil bei Verwendung von Block RAM besteht darin, dass Logikblöcke gespart werden. Der Nachteil ist, dass dieser in seiner Zugriffszeit langsamer ist. Damit eine Kommunikation mit der Außenwelt erfolgen kann, sind 500 I/O-Böcke mit dem Gehäuse des FPGAs verbunden.

Für die erste Umsetzung ist es empfehlenswert, einen FPGA mit reichlichen vorhandenen Ressourcen zu verwenden, damit die erste Implementierung nicht daran scheitert. Ist die Funktion gegeben und vollständig, kann mit der Optimierung des darauf laufenden Codes begonnen werden. Anhand des Kompiliervorgangs wird der Verbrauch von vorhanden Ressourcen auf dem FPGA angezeigt, sodass anschließend ein FPGA mit geeigneter Größe verwendet werden kann. Dies spielt keine allzu große Rolle, da die Umsetzung und Funktionsfähigkeit dieser GNSS-Störsignalunterdrückungseinheit im Vordergrund steht.

6.3 Anforderungen

Da die GNSS-Störsignalunterdrückungseinheit eine Reihe von Anforderungen erfüllen sollte, werden diese nun definiert. Hierbei wird spezifiziert, welche Funktionen bei der Verwendung als Front-End, Repeater und Störsignalunterdrückung erreicht werden sollen. Außerdem werden wichtige Eigenschaften und einstellbare Parameter der Bedienkonsole festgelegt. Zuletzt erfolgt noch eine Definition der Anforderungen der Algorithmen auf dem FPGA.

6.3.1 Funktion als GNSS-Front-End

Das USRP-RIO soll als Front-End mit Störsignalunterdrückungsfähigkeit für den am Institut vorhandenen Softwareempfänger dienen können. Es soll in der Lage sein, dem Softwareempfänger die GNSS-Signale in Echtzeit zur Verfügung zu stellen oder als digitale Samples für eine spätere Verarbeitung zu speichern. Der Algorithmus zur Störsignalunterdrückung sollte vorzugsweise im laufenden Betrieb hinzugeschalten oder ausgeschalten werden können.

6.3.2 Funktion als GNSS-Repeater

Das USRP-RIO soll neben der Funktion als Front-End auch als GNSS-Repeater dienen. Die empfangenen Signale sollen nach einer möglichen Störsignalunterdrückung über den zweiten Kanal des Transceivers ausgesendet werden, was einen Empfang des „bereinigten“ Signals mit einem alternativen GNSS-Empfänger ermöglicht. Nicht immer ist die Funktionsweise als Front-End für den Softwareempfänger gewünscht.

6.3.3 Bedienkonsole

Mit der Bedienkonsole sollen wichtige Parameter für die Digitalisierung der GNSS-Signale einstellbar sein. Neben den Grundeinstellungen für das USRP sollen Einstellungen für den Signalempfang vorhanden sein. Um eine Beurteilung des empfangenen Signals direkt im Front-End vornehmen zu können, werden notwendige Informationen von diesem bereits sichtbar gemacht. Das empfange Signale soll im Zeit- und Frequenzbereich angezeigt werden. Da eine Visualisierung in Echtzeit einen erheblichen Rechenaufwand auf dem Host-Rechner zur Folge hätte, sollten die Graphen mittels Tastendruck aktualisierbar sein.

6.3.4 Anforderungen an den FPGA-Code

Die auf dem FPGA zu realisierenden Algorithmen sollen jeweils als eigenständiges Modul in einem sogenannten „subVI“ realisiert werden. Dadurch kann während der Entwicklungsphase jeder Algorithmus auf seine korrekte Funktionsweise getestet werden. Sind die Module fertig programmiert und die Funktionsweise geprüft, müssen diese im Idealfall nur noch in der entsprechenden Reihenfolge zusammengefügt werden. Ein weiter Vorteil ist, dass diese fertigen Module später in anderen FPGA-Projekten mit LabVIEW einsetzbar sind.

Bevor die beiden Algorithmen in den Quellcode des Beispielprogrammes des USRP-RIO eingebunden werden, werden diese in einem eigenen Projekt umgesetzt. Somit kann die Implementierung modular erfolgen, was ein einfaches Testen sowie die Fehlersuche vereinfacht. Die Reihenfolge wird wie folgt festgelegt:

1. Neues Projekt mit jeweils einem Host-VI und FPGA-VI erstellen
2. Daten über FIFO auf FPGA geben und wieder abholen
3. Implementieren der I/Q-zu-ZF-Transformation
4. Implementieren des FFT-Unterdrückungsalgorithmus
5. Testen auf korrekte Arbeitsweise
6. Beide Algorithmen in das Beispielprogramm einbinden
7. Testlauf

Der FFT-Unterdrückungsalgorithmus soll Flexibilität besitzen. Es gilt, den FPGA-Code so zu programmieren, dass ohne Kompilieren verschiedene Parameter bei dem FFT-Unterdrückungsalgorithmus änderbar sind. Die Fensterfunktion soll für Testzwecke an- und abschaltbar sein. Ebenso sollen bei der späteren Mittelwertbestimmung für die Schwellwertbestimmung verschiedene Längen wählbar sein. Der gesamte Unterdrückungsalgorithmus soll aktiviert und deaktiviert werden können, um diesen bei nicht vorhandener Interferenz abzuschalten.

6.4 Vorhandene FFT-Bibliotheken für den FPGA

Die FFT ist der bekannteste und am häufigsten verwendete Algorithmus zur Berechnung der DFT in der digitalen Signalverarbeitung, weil er besonderes einfach zu implementieren und in vielen Anwendungsfällen die beste Effizienz bietet [29]. Das Bearbeiten von Signalen lässt sich bei einigen Funktionen besser im Frequenzbereich als im Zeitbereich durchführen, wie zum Beispiel das Entfernen fehlerhafter Frequenzen.

Weil der Aufwand für das Implementieren einer FFT sehr groß ist, stehen sogenannte „Bibliotheken“ zur Verfügung. Ein weiterer Grund für die Verwendung von Bibliotheken besteht darin, dass die FFT auf einem FPGA eine hohe Anzahl der darauf zur Verfügung stehenden Ressourcen benötigt. Ein vorgefertigtes Modul ist effizient programmiert. In LabVIEW stehen drei verschiedene Typen von vorgefertigten FFT-Funktionen zur Verfügung, welche in den untergliedernden Kapiteln erläutert sind. Für die Implementierung der beiden Algorithmen (I/Q-zu-ZF-Transformation und FFT-Unterdrückungsalgorithmus) wurde die LabVIEW eigene FFT gewählt.

6.4.1 LabVIEW FPGA FFT

Die LabVIEW FPGA FFT ist ein von NI bereitgestelltes Express-VI. Dieses befindet sich in der „FPGA Math & Analysis“ Toolpalette von LabVIEW. Mit dieser Funktion wird die FFT auf dem FPGA umgesetzt. Es können Parameter und Ausführungsarten der FFT auf dem FPGA mittels eines Konfigurationsfensters eingestellt werden. Einstellbar ist eine Länge von 8 bis 8192, sowie weitere Parameter über die Ausführung auf dem FPGA. Es kann die Formatierung des Ausgangsdatentypen formatiert, der Ausführungsmodus, die Taktrate und der Datendurchsatz definiert werden. Dies ist in Abbildung 35 dargestellt. Werden Einstellungen innerhalb des Konfigurationsfensters geändert, ist ein erneutes Kompilieren des gesamten FPGA-Codes erforderlich.

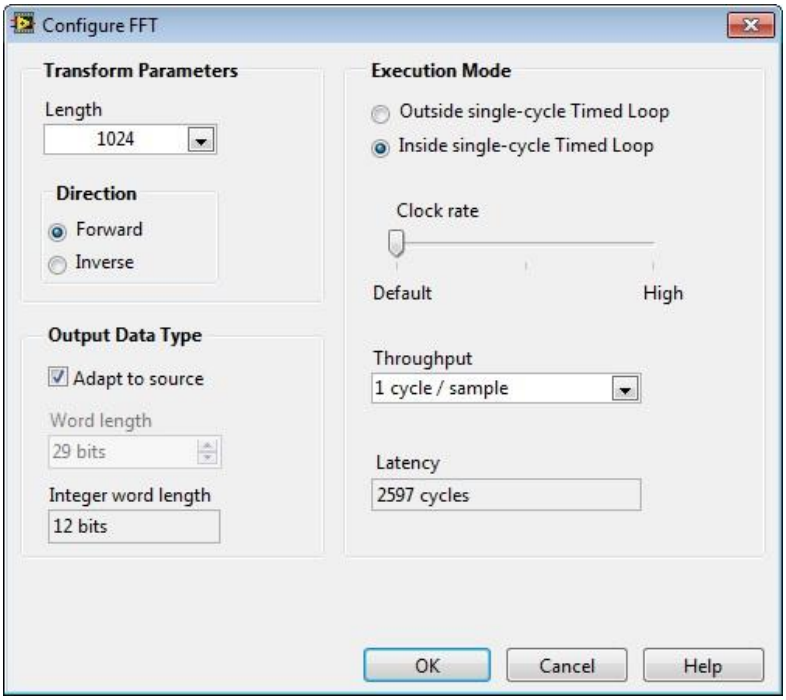


Abbildung 35: Konfigurationsfenster LabVIEW FPGA FFT

Als Parameter wird eine FFT mit der Länge von 1024 gewählt, da sich bei der Unterdrückung von Störsignalen in Kapitel 5 der beste Unterdrückungserfolg mit dieser Länge herausgestellt hat.

6.4.2 Xilinx FFT

Der FFT-Core von Xilinx kann eine FFT von 8 bis 65536 Punkten oder eine inverse komplexe Transformation von bis zu 12 parallelen Kanälen berechnen. Die Eingangsdatei ist ein Vektor von komplexen Werten, dargestellt als Zweierkomplement im Bereich von 8 bis 34 Bit Breite oder als Gleitkommazahlen mit einer Breite von 32 Bit. Die Phasenfaktoren können 8 bis 34 Bit breit sein. Alle Speicher sind auf dem Chip als Verwendung mit Block RAM oder verteilter RAM realisiert. Drei arithmetische Typen stehen zur Verfügung: Vollste Genauigkeit (nicht skaliert), skalierte Festkomma und Block-Gleitkomma. Mehrere Parameter sind für die Laufzeit der FFT konfigurierbar: Die Punktgröße, die Wahl der vorwärts oder inversen Transformation und die Skalierung. Vier Architekturen sehen zur Verfügung, um einen Kompromiss zwischen der Größe und der Aufführungszeit bereitzustellen [37]. Dieses vorgefertige FFT bietet viele Konfigurationsmöglichkeiten und eine hohe Performance.

6.4.3 Xilinx LTE FFT

Der LTE FFT-Core von Xilinx verfügt über alle Transformationslängen, die für 3GPP LTE Spezifikationen benötigt werden, inklusive der 1536-Punkt Transformation für eine Bandbreite von 15 MHz. Die Transformationslänge, Transformationsrichtung, zyklische Präfix-Länge und der Skalierungszeitplan können auf Frame-Basis konfiguriert werden [38]. Da die Art der Xilinx LTE FFT auf Mobilfunkanwendungen spezialisiert ist und die Relevanz im GNSS-Bereich liegt, wird diese nicht weiter erläutert.

6.5 Grundlagen der LabVIEW FPGA Programmierung

In diesem Teil werden die Grundlagen für die Programmierung eines FPGAs mit LabVIEW erläutert. Da auf einem FPGA sehr häufig mit dem Datentyp „Festkommazahl“ (engl.: Fixed-Point Data; Abk.: FXP) gearbeitet wird, erfolgt für diesen Datentyp eine Erklärung. Außerdem werden auch die notwendigen Reduzierungen der Datenbreite nach einer arithmetischen Operation ermittelt. Ist die Festkommazahl behandelt, wird auf den Speicher des FPGAs eingegangen. Dieser wird bei der Implementierung von Algorithmen auf dem FPGA und bei dem Datentransfer auf dem Host-Rechner verwendet. Bei der digitalen Signalverarbeitung auf einem FPGA müssen die Prinzipien der Datenflusssteuerung eingehalten werden, deshalb erfolgt darüber eine Beschreibung. Ebenso erfolgt eine Erläuterung der Datenerfassungseinheit des gegebenen FPGA-Codes.

6.5.1 Festkommazahl

Der große Vorteil bei der Verwendung von Festkommazahlen besteht darin, dass bei der Berechnung die Flexibilität von Fließkommazahlen mit niedrigerem Rechenaufwand erreicht wird. Bestimmte arithmetische Operationen lassen sich durch Schiebeoperationen realisieren. Es ist zu erwähnen, dass nach bestimmten arithmetischen Operationen eine höhere Datenbreite der FXP vorhanden ist. Diese müssen wieder gekürzt werden, wodurch die Genauigkeit abnimmt [39].

Im I/O-Node (Input/Output) des FPGAs werden die Daten in der Integer-Arithmetik bedingt durch den Transceiver des USRP-RIOs bereitgestellt. Diese werden gleich nach dem Erhalt in das Festkommazahlenformat für die weitere Verarbeitung umgewandelt und in einem Cluster für den Weitertransport gebündelt. Abbildung 36 stellt diesen Vorgang in LabVIEW dar.

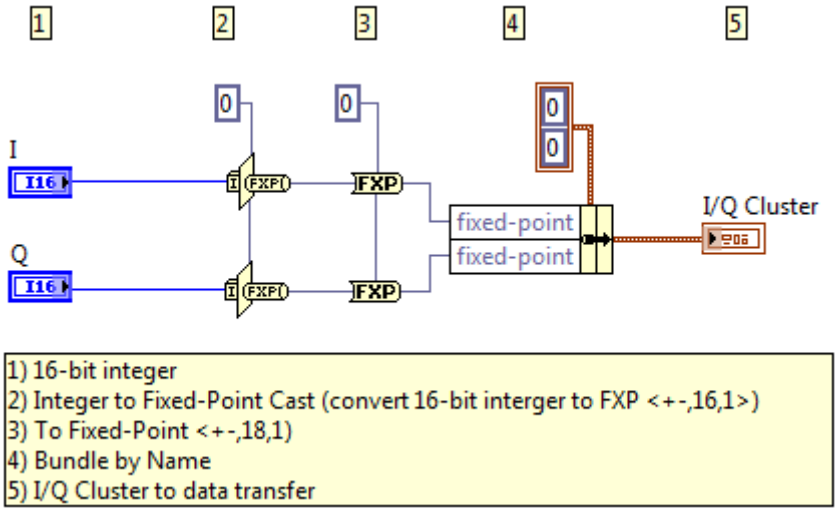


Abbildung 36: Konvertierung der Integer-Arithmetik in die FXP-Arithmetik

Die erste Station nach dem Empfang der Daten auf dem FPGA ist das „Integer to Fixed-Point Cast“ Element. Dieses skaliert den Integer-Wert auf den gewünschten FXP-Wert. Diese Funktion entspricht einer arithmetischen Verschiebung des Eingangs mit einer konfigurierbaren Überlaufhandhabung, so dass das letzte und niederwertigste Bit der Eingabe das niederwertigste Bit der Ausgabe wird. Abschließend wird der Wert mit dem „To Fixed-Point“ auf 18 Bit skaliert.

Nach bestimmten arithmetischen Operationen vergrößert sich der Wertebereich des FXP. Es wird mit einer Datenbreite von 18 Bit gearbeitet. Die Darstellung erfolgt mit folgender Formatierung: <±,18,1>. Das erste Feld gibt die Vorzeichenkodierung an. Das zweite Feld gibt die gesamte Datenbreite an und das dritte Element repräsentiert die Anzahl der Bits vor dem Komma. Nach einer FFT mit einer Länge von 1024 ergibt sich auf dem FPGA eine Ausgangsgröße von <±,26,12>. Es gilt, diese wieder auf den Wertebereich des Eingangs zu reduzieren. Wenn mehrere aufeinander folgende Operationen auszuführen sind, würde der Eingangswert innerhalb kürzester Zeit sein erlaubtes Maximum von 64 Bit erreichen. Außerdem ist es in der FFT-Funktion nicht erlaubt, Eingangswerte über 24 Bit einzuspeisen. Ein weiter Grund, um den Datentyp klein zu halten ist, dass mit steigender Bit-Größe der Ressourcenverbrauch auf dem FPGA zunimmt. Wie in Abbildung 37 zu erkennen ist, sind zur Reduzierung mehrere Schritte notwendig.

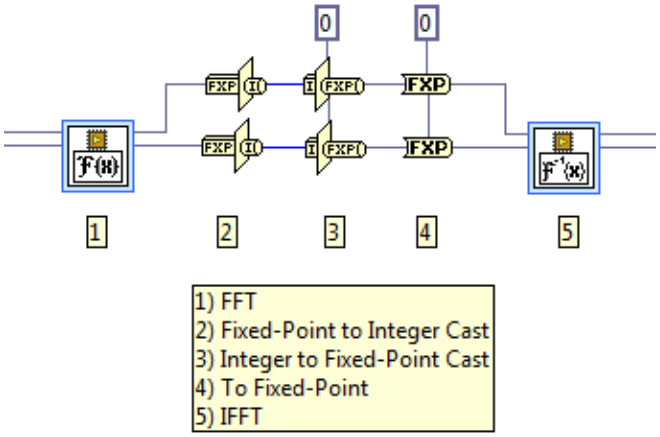


Abbildung 37: Reduzieren der FXP-Datenbreite

Zunächst wird der FXP-Wert mittels der „Fixed-Point to Integer Cast“ Funktion in den Integer-Datentyp umgewandelt. Die Formatierung des Integer-Wertes wird von LabVIEW automatisch so gewählt, dass der Eingangsbereich darin dargestellt werden kann. Steht ein FXP-Wert mit maximal 16 Bit am Eingang, wird I16 gewählt. Da nach der FFT mit der Länge von 1024 eine Ausgangsgröße des FXP von 26 Bit vorhanden ist, wird automatisch ein I32 Bit-Integer gewählt. Den Integer-Wert gilt es nun wieder in FXP umzuwandeln und überschüssige Bits zu entfernen. Bei den Einstellungen in dieser Funktion ist als Rundungsmodus „Truncate“ gewählt, welcher das „Abschneiden“ der überflüssigen Bits bewirkt. Die Reduzierung von fraktionellen Bits schont Ressourcen des FPGAs [40] [45].

Um die Konfiguration und Ermittlung der Werte für die FXP-Reduzierung nicht jedes Mal neu bestimmen, wurden diese in der Tabelle 15 und Tabelle 16 notiert. Tabelle 15 zeigt die Einstellungen der notwendigen Bausteine für die Reduzierung des FXP bei der I/Q-zu-ZF-Transformation. Die Reduzierung muss nach der FFT und nach der inversen FFT erfolgen.

Tabelle 15: Reduzierung des FXP bei der I/Q-zu-ZF-Transformation

	FFT (Länge: 128)	**IFFT (Länge: 256)**
Eingangsgröße (FXP)	<±,18,1>	<±,18,1>
Ausgangsgröße (FXP)	<±,26,9>	<±,27,2>
Integer to FXP	<±,32,6> Rounding Mode: Truncate Overflow Mode: Wrap	<±,32,16> Rounding Mode: Truncate Overflow Mode: Wrap
To FXP	<±,18,1> Rounding Mode: Round Half-Even Overflow Mode: Saturate	<±,18,1> Rounding Mode: Round Half-Even Overflow Mode: Saturate

Tabelle 16 zeigt die Einstellungen der notwendigen Elemente für die Reduzierung des FXP bei dem FFT-Unterdrückungsalgorithmus. Hierbei erfolgte eine Reduzierung nach der Fensterfunktion, der FFT und der inversen FFT.

Tabelle 16: Reduzierung des FXP bei dem FFT-Unterdrückungsalgorithmus

	Fensterfunktion	**FFT (Länge: 1024)**	**IFFT (Länge: 1024)**
Eingangsgröße (FXP)	<±,18,1>	<±,18,1>	<±,18,1>
Ausgangsgröße (FXP)	<±,20,2>	<±,29,12>	<±,29,2>
Integer to FXP	<±,32,13> Rounding Mode: Truncate Overflow Mode: Wrap	<±,32,4> Rounding Mode: Truncate Overflow Mode: Wrap	<±,32,16> Rounding Mode: Truncate Overflow Mode: Wrap
To FXP	<±,18,1> Rounding Mode: Round Half-Even Overflow Mode: Saturate	<±,18,1> Rounding Mode: Round Half-Even Overflow Mode: Saturate	<±,18,1> Rounding Mode: Round Half-Even Overflow Mode: Saturate

Nach dem Vorgang zur Reduzierung der Datenbreite muss gewährleistet sein, dass der gesamte Zahlenraum mit einen I16 Wert von -32768 bis 32767 abgedeckt wird. Um die richtige Funktionsweise der Reduzierung des FXP-Wertes zu testen, wurde ein simuliertes Sinussignal mit einer Amplitude von minimaler bis maximaler möglicher Aussteuerung eingespeist. Je kleiner der Eingangswert des Integer wird, desto mehr Artefakte sind zu erkennen, da die Rundungsgenauigkeit abnimmt. Es ist grundsätzlich immer mit gewissen Verlusten an der Genauigkeit zu rechnen. Diese sind nicht vermeidbar, werden aber durch die Vorteile dieses Datentyps in Kauf genommen.

6.5.2 Speicher auf dem FPGA

Bei speziellen Funktionen auf dem FPGA ist es notwendig, Daten vorübergehend zu speichern. Dies kann mit verteilten RAM (generiert aus Logikzellen), dem internen Block RAM oder dem externen DRAM realisiert werden. Die Implementierung des Speichers kann als Variable, FIFO (First-in first-out) oder Memory erfolgen. Diese Speichermethoden werden in der FPGA-Programmierung für zeitkritische Aufgaben, zur Manipulation von Daten oder für die DMA-Übertragung (Direct Memory Access) benötigt.

Werden zum Speichern auf dem FPGA nur einzelne Elemente benötigt, sollte man auf lokale oder globale Variablen zurückgreifen. Soll eine Kommunikation zwischen dem FPGA und dem Host-Rechner erfolgen, verwendet man Register. Dies wird auf dem Host-Code mit dem „Read/Write Control“ realisiert. Mit diesem können Steuerbefehle und Datenwerte von dem Host-Rechner an den FPGA übergeben werden oder Statusabfragen des FPGAs vom Host-Rechner erfolgen.

Bei einem FIFO handelt es sich um eine Speichermethode, bei der zuerst geschriebene Daten auch zuerst ausgelesen werden. Die Daten lassen sich somit puffern. Vor bestimmten Algorithmen treten Latenzzeiten auf, so dass die Daten dafür bis zur nächstmöglichen Bearbeitung gehalten werden müssen. FIFOs lassen sich mit drei verschiedenen Methoden auf dem FPGA implementieren: Als Flip-Flops, LUTs und Block RAM. Jede Möglichkeit bietet Vor- und Nachteile. Bei der Implementierung als geteilter RAM in Form von Flip-Flops oder LUTs werden schnelle Zugriffszeiten realisiert. Der Speicherplatz ist bei dieser Methode beschränkt und es wird auf begrenzte Logikzellen des FPGAs zugegriffen. Der Block RAM bietet mehr Speicherplatz, ist aber in seiner Zugriffszeit langsamer.

Es gibt drei Typen von FIFOs: Der erste Typ ist die Verwendung als „Target-Scoped“. Mit diesem Typ kann nur ein Datentransfer innerhalb des FPGA erzeugt werden. Soll ein Datentransport zwischen Host-Rechner und FPGA erfolgen, wird der DMA-Typ gewählt. Dieser lässt sich nur als Block RAM implementieren. Um Daten über die PXIe-Schnittstelle unter verschiedenen NI-Geräten auszutauschen gibt es noch das Peer-to-Peer-FIFO. Das FIFO als „DMA Target-to-Host“ wird für den Datentransfer zwischen dem Host-Rechner und dem FPGA eingesetzt. Diese Zugriffsverfahren greift über das Bussystem direkt auf den Speicher des FPGA und Host-Rechner zu. Standardmäßig ist das FIFO auf der Seite des FPGAs mit 1024 Elementen vorkonfiguriert. Auf der Seite des Host-Rechners sind standardmäßig 10.000 Elemente belegt. Mit einem 32 Bit großen Integer-Datentyp wird bei 10.000 Elementen auf dem Host-Rechner ein Arbeitsspeicher mit einer Größe von 40k Byte allokiert. Auf dem FPGA werden 4096 Byte allokiert.

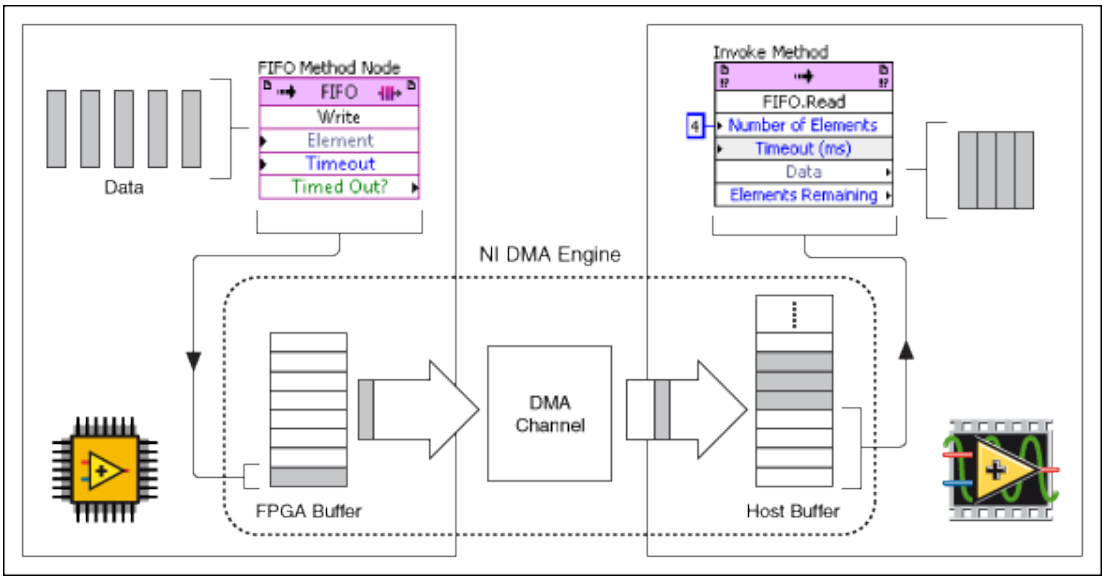


Abbildung 38: Funktionsweise des DMA bei NI FPGA (Quelle: [41])

Wird die Reihenfolge der Daten auf dem FPGA manipuliert oder ist die Speichergröße des FIFOs nicht ausreichend, muss auf ein Memory zurückgegriffen werden. Ein Memory verfügt über einen zusätzlichen Anschluss für die Adressverwaltung. Dies stellt einen erhöhten Aufwand bei der Umsetzung dar. Somit können Daten in beliebiger Reihenfolge gelesen und ausgegeben werden. Zusätzlich bietet der Memory neben der

Implementierung als LUT oder Block RAM die Möglichkeit, den verbauten DRAM des USRP-RIOs zu verwenden. Mit diesem sind höhere Speicherkapazitäten möglich, was jedoch langsamere Zugriffszeiten zur Folge hat.

6.5.3 Datenflusssteuerung

Die Datenflusssteuerung wird in NI LabVIEW FPGA „Handshaking“ genannt. Dies ist eine elementare Methode, um den Ablauf der Punkt-zu-Punkt-Signalverarbeitung unter den verschiedenen Modulen im FPGA mit LabVIEW in einer SCTL (Single Cycle Timed Loop) zu synchronisieren und zu steuern. Jedes Modul arbeitet mit einer unterschiedlichen Verarbeitungsgeschwindigkeit, abhängig von der Latenzzeit und dem Datendurchsatz. Um keine Verletzung des Datenflusses zu erzeugen, müssen die Regeln des Handshakings eingehalten werden. Ist das nicht der Fall, gehen Samples verloren. Bei jedem umzusetzenden Algorithmus wird mit dem 4-wire Handshaking gearbeitet. Dies bedeutet, dass jedes Modul über zwei boolesche ansteuerbare Eingänge und zwei boolesche Ausgänge zur Kommunikation mit dem vorhergehenden beziehungsweise nachfolgenden Modul verfügen. In Tabelle 17 sind diese mit deren Funktionsbeschreibung aufgelistet.

Tabelle 17: Wichtige Funktionen des 4-wire Handshaking

Anschlussname	**Funktion**
Input valid (Eingangswert gültig)	Spezifiziert, dass der nächste Eingangswert für das Prozessieren ankommt und gültig ist.
Ready for output (Bereit für Ausgabe)	Spezifiziert, dass das nachfolgende Modul gültige Werte empfangen kann.
Output valid (Ausgangswert gültig)	Zeigt an, dass die aktuellen Ausgangsdaten für das nachfolgende Modul gültig sind.
Ready for input (Bereit für Eingabe)	Zeigt an, dass das Modul bereit ist, um neue Eingangswerte anzunehmen.

In Abbildung 39 ist der Algorithmus der I/Q-zu-ZF-Transformation dargestellt, welcher das Verhalten des 4-wire Handshaking zeigt. Produziert das vorherige Modul noch kein gültiges Ergebnis, ist das Nachfolgende gezwungen darauf zu warten. Sind gültige Ausgangswerte vorhanden, wird das durch TRUE am „Output valid“ signalisiert und an den „Input valid“ des Folgemoduls weitergegeben. Damit dieses überhaupt einen Wert ausgeben darf, muss das nachfolgende Modul mit „ready for input“ signalisieren, dass Daten empfangen werden können und über ein Feedback-Node an das vorherige „ready for output“ geben. Ist das nachfolgende Modul nicht in der Lage, Werte zu empfangen, darf das vorherige keine Daten ausgeben. Wenn das 4-wire Handshaking nicht eingehalten wird, gehen Samples verloren. Nach diesem Prinzip ist das letzte Modul in der Kette das Hauptglied zur Steuerung.

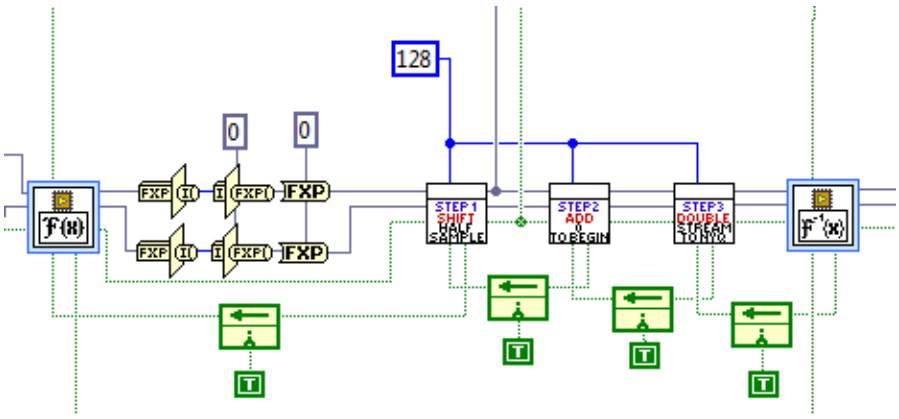


Abbildung 39: Prinzip des 4-wire Handshaking

6.5.4 Datenerfassungseinheit

Die bereits vorhandene Datenerfassungseinheit nimmt die empfangenen Signale nach dem AD-Wandler auf. Standardmäßig werden hierbei drei Schritte durchlaufen, welche das Signal digital im FPGA aufbereiten. Dies geschieht mit dem digitalen „Downconverter“, welcher aus drei Modulen besteht:

1. **I/Q Impairments**
   Modifiziert die I/Q-Daten, um die Signalbeeinträchtigung zu übernehmen.
2. **Frequency Shift**
   Wendet eine digitale Frequenzverschiebung auf die I/Q-Daten an. Ein numerisch gesteuerter Oszillator (engl.: Numerically Controlled Oscillator; Abk.: NCO) erzeugt ein Kosinus/Sinus-Paar, welches eine komplexe Multiplikation mit den I/Q-Daten durchführt. Der reine Effekt ist das Verschieben des komplexen Spektrums zur linken oder rechten Seite im Frequenzbereich.
3. **Fractional Decimator**
   Dezimiert das Eingangssignal auf die gewünschte Abtastrate unter Anwendung einer FIR-Filter Implementierung.

6.6 Implementieren der Algorithmen

In diesem Kapitel erfolgt die Implementierung der gegebenen Algorithmen unter Einhaltung der angegebenen Anforderungen aus Kapitel 6.3.4. Zuerst wurde die FPGA-Infrastruktur für das Implementieren der beiden Algorithmen geschaffen, in welcher diese auch getestet wurden.

6.6.1 Erstellen der FPGA-Infrastruktur

Im neu angelegten FPGA-Projekt wurde ein Host-VI und ein FPGA-VI erstellt. Beide sind leer und beinhalten keinen LabVIEW-Code. Das Host-VI wird auf dem Rechner ausgeführt und kann mit dem FPGA über eine „Read/Write-Control“ kommunizieren. Über ein DMA-FIFO erfolgt der Datenaustausch zwischen Host-Rechner und FPGA.

Im Host-VI wurden zwei Testsignale bereitgestellt. Das Erste ist ein Array bestehend aus acht Elementen. Als zweites Testsignal wurde auf dem Host-Rechner eine kontinuierliche

Schwingung als I/Q-Signal generiert. Der Datentyp beider Testsignale ist ein 16 Bit Integer. Wahlweise kann eines der Testsignale ausgewählt werden. Für das Implementieren des Algorithmus wurde bewusst am Anfang der Integer-Datentyp gewählt, da im Gegenzug zur eigentlich benötigten FXP ganze Zahlen behandelt werden. Damit ist es einfacher und übersichtlicher die richtige Arbeitsweise auf dem FGPA zu überprüfen.

Im LabVIEW FPGA-Projekt gibt es vier Möglichkeiten den FPGA-Code auszuführen: Auf dem FPGA, auf dem Entwicklungsrechner mit simulierten I/O oder realen I/O oder auf einem „Third-Party Simulator“. Um den Code auf dem FPGA zu betreiben, ist vorher eine Kompilierung notwendig. Um dies zu vermeiden wurde die zweite Variante, das Simulieren auf dem Entwicklungsrechner, gewählt. Für das Überprüfen der Funktionalität reicht dies aus. Ist eine fehlerfreie Funktionsweise gegeben, wird der FPGA-Code kompiliert und überprüft.

Die Überprüfung der Infrastruktur zum Implementieren der eigenen Algorithmen wurde wie folgt durchgeführt: Es wurden nach der Reihe beide Testsignale zunächst auf den FPGA und anschließend wieder auf den Host-Rechner gegeben. Sind die auf dem Host-Rechner erzeugten Daten mit den über den FPGA gesendeten Daten identisch, ist die Arbeitsweise korrekt. Sobald das funktionsfähige Grundgerüst steht kann mit der Implementierung der Algorithmen auf dem FPGA begonnen werden.

6.6.2 I/Q-zu-ZF-Transformation

6.6.2.1 Funktionsweise

Da das USRP-RIO das empfangene Signal im Basisband als I/Q-Signal (komplexes Signal) bereitstellt, aber der Softwareempfänger ein reelles Signal mit einer Zwischenfrequenz (ZF) benötigt, muss dieses vor der Verwendung umgewandelt werden. Anschließend wandelt der Softwareempfänger das Signal wieder in das Basisband um und nimmt die benötigten Berechnungen für das Bestimmen der Position vor. Das stellt einen doppelten Aufwand dar, ist aber aufgrund der aktuellen Ausgangslage nicht anders realisierbar. In Abbildung 40 ist der Algorithmus der I/Q-zu-ZF-Transformation auf einem Rechner dargestellt, welchen es auf dem FPGA des USRP-RIO zu implementieren gilt.

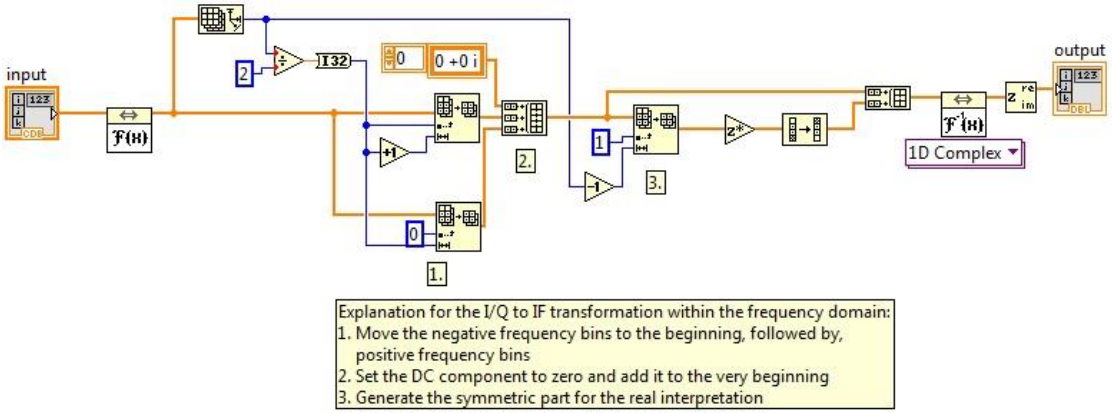


Abbildung 40: I/Q-zu-ZF-Transformation (Quelle: [42])

6.6.2.2 Implementieren auf dem FPGA

Zur einfacheren Implementierung auf dem FPGA erfolgte eine Aufteilung des Algorithmus in drei Teile. Für jeden dieser Schritte wurde ein eigenes subVI erstellt und mit „step1“, „step2“ und „step3“ benannt. Anschließend wurde eine Case-Struktur mit drei Ebenen erzeugt. In jeder Ebene wurde ein Schritt umgesetzt und überprüft.

Im **ersten Schritt** werden die Samples von der Hälfte der Fensterlänge vertauscht. Abbildung 41 stellt diesen Vorgang grafisch mit einer Fensterlänge von 8 dar. Bei einer Fensterlänge von 128, wie der Algorithmus später verwendet wird, beträgt der Datendurchsatz 1 Zyklus/Sample und die Latenzzeit beträgt 64 Zyklen. Bei diesem Algorithmus entspricht der Datendurchsatz die Anzahl der Ausgangswerte dividiert durch die Anzahl der Eingangswerte.

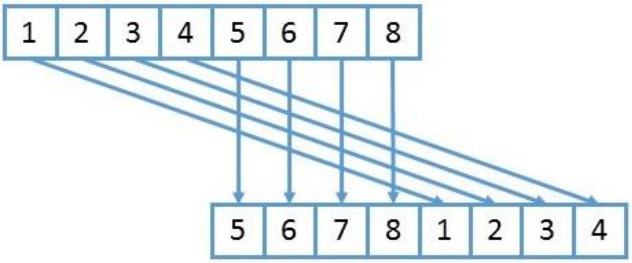


Abbildung 41: Schritt 1 der I/Q-zu-ZF-Transformation

Im **zweiten Schritt** wird den Ausgangswerten in Abhängigkeit der Fensterlänge eine 0 voran gesetzt. Abbildung 42 stellt diesen Vorgang grafisch dar. Somit ist der Ausgangswert immer um einen Wert höher als dessen Eingangswert, so dass immer ein zusätzlicher Zyklus benötigt wird. Dadurch ergibt sich bei einer Fensterlänge von 128 ein Datendurchsatz von 1,0079 Zyklen/Sample und die Latenzzeit beträgt 1 Zyklus.

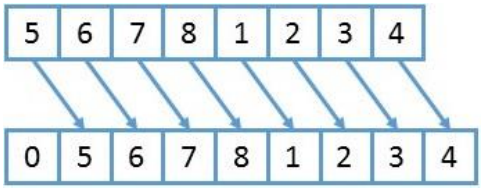


Abbildung 42: Schritt 2 der I/Q-zu-ZF-Transformation

Im **dritten Schritt** werden die ersten neun Eingangswerte so ausgegeben, wie sie eingegeben wurden. Danach erfolgt eine Spiegelung der zweiten Hälfte. Im Q-Kanal wird das Vorzeichen im gespiegelten Teil inkrementiert. Das Spiegeln wird mithilfe von Speichern gelöst. Mittels eines Vorwärtszählers werden die Eingangswerte in den Speicher geschrieben und mit einem Rückwärtszähler wieder ausgelesen. Während des Lesevorgangs dürfen keine neuen Werte in das Modul eingegeben werden. Die vorhergehenden Module müssen warten, bis der Speicher vollständig gelesen ist. Die Speichertiefe entspricht der Fensterlänge. Bei einer Fensterlänge von 128 beträgt der Datendurchsatz 1,9845 Zyklen/Sample und die Latenzzeit beträgt 0 Zyklen.

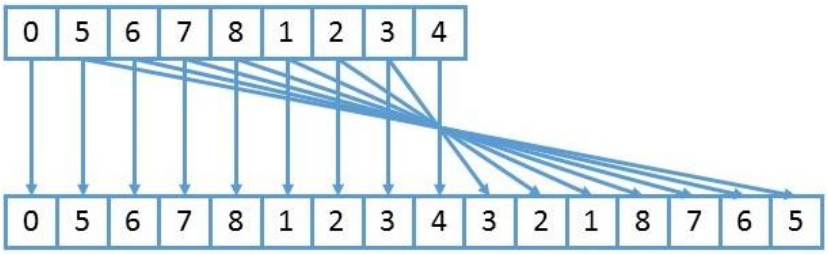


Abbildung 43: Schritt 3 der I/Q-zu-ZF-Transformation

Zum Testen der drei Schritte wurde der gegebene Algorithmus auf dem Host-Rechner mit dem auf dem FPGA implementierten Code parallel ausgeführt. Zur visuellen Überprüfung der korrekten Arbeitsweise wurden die Ausgangswerte beider Algorithmen auf dem Host-Rechner verglichen. War das Ergebnis nicht identisch, ist der Code auf dem FPGA fehlerbehaftet. Zur Überprüfung des Codes gibt es die Möglichkeit, während des Testbetriebs im FPGA ein Sampling Probe auf eine oder mehrere Leitungen zu setzen, welches den aktuellen Wert in Abhängigkeit des Takts anzeigt. In Abbildung 44 ist die Sampling Probe von Schritt 2 dargestellt, welche die fehlerfreie Arbeitsweise zeigt. Das Überprüfen mit der Sampling Probe wurde bei jedem Schritt angewendet. Wenn alle drei Schritte korrekt arbeiten, werden diese hintereinander geschaltet. Zur Finalisierung fehlen noch eine FFT am Anfang und eine IFFT am Ende. Die FFT-Länge wurde mit 128 und die IFFT-Länge mit 256 gewählt. Die Länge der IFFT muss aufgrund der I/Q-zu-ZF-Transformation doppelt so hoch sein wie die der FFT. Die Fensterlänge der einzelnen Schritte wurde dann auf eine Länge von 128 angepasst. Wie in Kapitel 6.4 beschrieben, bietet LabVIEW für die FFT ein vorgefertigtes Modul. Die FFT, IFFT und die drei Schritte wurden unter Einhaltung des 4-wire Handshaking zusammengefügt. Abbildung 39 aus Kapitel 6.5.3 zeigt den kompletten Algorithmus der I/Q-zu-ZF-Transformation.

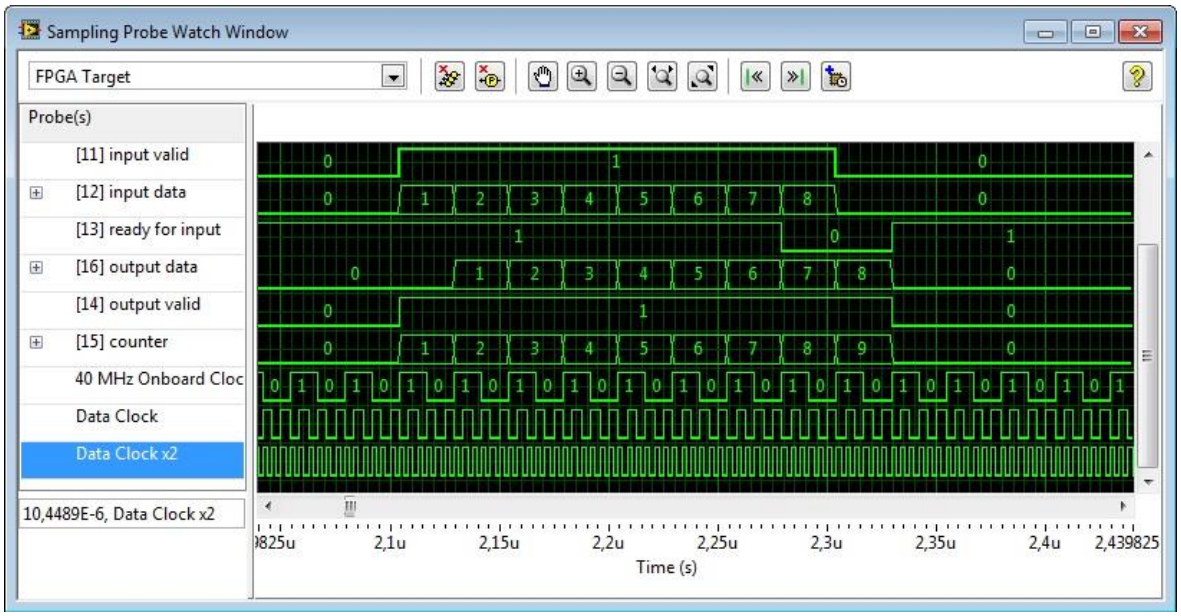


Abbildung 44: Sampling Probe von Schritt 2

Um den nun fertigen Algorithmus auf seine Fehlerfreiheit zu überprüfen, wurde das simulierte I/Q-Signal auf den FPGA gegeben. Es muss sichergestellt sein, dass dieser mit

einer minimalen und maximalen Amplitude arbeitet. Die richtige Handhabung der Festkommazahl, beschrieben in Kapitel 6.5.1, gewährleistet dies. Wenn alle Anforderungen erfüllt sind, kann aus allen Modulen ein eigenes subVI erstellt werden, was den fertigen Algorithmus zur I/Q-zu-ZF-Transformation für den FPGA darstellt. Dieses neu erzeugte Modul wurde anschließend dem Beispielprogramm hinzugefügt.

Um bei einer späteren Verwendung des Algorithmus eine Aussage über Latenzzeit und Datendurchsatz zu erhalten, wurde dies in der Beschreibung eingefügt. Der gesamte Datendurchsatz des Ablaufs entspricht dem höchsten vorkommenden Wert in der Kette. Die Latenzzeit summiert sich aus allen Modulen einer Kette.

Tabelle 18: Datendurchsatz und Latenzzeit der I/Q-zu-ZF-Transformation

Schritt	**Datendurchsatz [Zyklen/Sample]**	**Latenzzeit [Zyklen]**
FFT (128)	1	347
Step1	1	64
Step2	1,0079	1
Step3	1,9845	0
IFFT (256)	1	668
Gesamt:	1,9845 (~2)	1080

6.6.3 FFT-Unterdrückungsalgorithmus

6.6.3.1 Funktionsweise

Eine Möglichkeit zur Unterdrückung eines Störsignals ist, das Signal mit einer FFT in seine spektralen Teile zu zerlegen. Anhand des Frequenzbereichs lassen sich die darin enthaltenen Störkomponenten durch eine Schwellwertbestimmung feststellen und entfernen. Dies bewirkt ein Auslöschen der fehlerhaften Frequenzen. Danach gilt es, das Signal mittels einer IFFT wieder in seinen Zeitbereich zu transformieren. Diesen Vorgang zeigt Abbildung 45. Eine detailliertere Beschreibung dieses Algorithmus erfolgte in Kapitel 5.4.1.

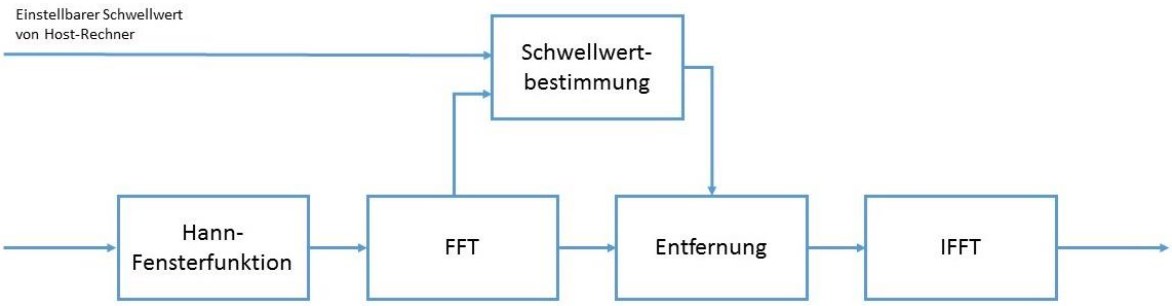


Abbildung 45: Schematischer Ablaufplan des FFT-Unterdrückungsalgorithmus

6.6.3.2 Implementieren auf dem FPGA

Die FFT-Methode zur Störsignalunterdrückung benötigt ebenfalls wie die I/Q-zu-ZF-Transformation eine FFT und IFFT. Wie in Kapitel 6.4 erwähnt, ist die FFT ein sehr ressourcenverbrauchender Algorithmus auf dem FPGA. Deshalb kann in Betracht gezogen werden, direkt die benötigte FFT und IFFT von der I/Q-zu-ZF-Transformation für den FFT-Unterdrückungsalgorithmus zu verwenden. Da es sich bei beiden Algorithmen im Prinzip um eigenständige Funktionen handelt, wurden diese auch eigenständig implementiert. Ist es erwünscht, dass die Signale wieder über die Sendeeinheit ausgegeben werden, entfällt die I/Q-zu-ZF-Transformation.

Am Anfang des FFT-Unterdrückungsalgorithmus befindet sich eine Case-Struktur, mit welcher die Hann-Fensterfunktion aktiviert und deaktiviert werden kann. Ist die FFT erfolgt, muss die Schwellwertbestimmung durchgeführt werden. Um den dafür notwendigen Mittelwert zu bestimmen, ist es erforderlich, dass die I- und Q-Daten in der Polarform bereit stehen. Die Umwandlung erfolgte mit dem „High Throuput Rectangular To Polar“ Funktion aus der LabVIEW Toolpalette. Das erforderliche Modul zur Mittelwertbildung wurde ebenfalls in einer Case-Struktur verwendet. Diese Case-Struktur besteht aus vier Ebenen, in welcher ein fest definierter Wert für den Schwellwert gegeben ist und drei Mittelwertbestimmungen mit unterschiedlichen Längen. Gewählt wurden Längen von 512, 1024 und 2048 Samples. Der errechnete Mittelwert wird mit einem über den Host-Rechner einstellbaren Schwellwert multipliziert. Danach erfolgt ein Vergleich des Polarwertes mit dem multiplizierten Wert. Ist eine Störung vorhanden, zeigt sich dies durch einen höheren Polarwert. Der fehlerhafte Frequenzanteil wird durch eine Case-Struktur auf Null gesetzt und damit ausgelöscht.

Da die „Rectangular To Polar“ Funktion eine Latenzzeit von 21 Zyklen aufweist, musste der Hauptpfad mit dem gleichen Verzug versehen werden, um eine Auslöschung des richtigen Frequenzbereichs zu bewirken. Um eine Verzögerung herbeizuführen, diente eine „Discrete Delay“ Funktion. Nach der Störsignalauslöschung wurde das Signal mit einer IFFT in den Zeitbereich zurück transformiert. In Tabelle 19 ist der Datendurchsatz und die Latenzzeit des FFT-Unterdrückungsalgorithmus aufgelistet.

Tabelle 19: Datendurchsatz und Latenzzeit des FFT-Unterdrückungsalgorithmus

Schritte	**Datendurchsatz [Zyklen/Sample]**	**Latenzzeit [Zyklen]**
Fensterfunktion	1	3
FFT (1024)	1	2597
erzwungene Verzögerung	1	21
IFFT (1024)	1	2597
Gesamt:	1	5218

6.7 Kompiliervorgang

6.7.1 Grundlagen der Kompilierung

Kompilieren ist der Vorgang, bei welchem der LabVIEW FPGA-Code mit mehreren Schritten in eine Bit-Datei umgewandelt wird. Die Dauer beträgt ja nach Größe des vorhandenen Codes ein paar Minuten bis hin zu mehreren Stunden. Dies ist ebenso abhängig von der Leistung des Rechners. Der Code kann mit entsprechender Software auf dem eigenen Rechner, auf einem eigenen aufgesetzten Kompilierung-Server oder mit dem zahlungspflichtigen Dienst „Compile Cloud Service“ von NI kompiliert werden. Letztere bieten einen schnelleren Kompilierungsvorgang. Für die GNSS-Störsignalunterdrückungseinheit wurde die Kompilierung auf dem eigenen Rechner gewählt. In Abbildung 46 ist der schematische Ablauf des Kompilierungsvorgangs dargestellt.

Abbildung 46: Schematischer Ablauf des Kompilierungsprozesses (Quelle: [43])

Im ersten Schritt des Kompilierungsprozesses wird der LabVIEW FPGA-Code in eine Hardwarebeschreibungssprache (engl.: Hardware Description Language; Abk.: HDL) konvertiert. Die daraus resultierenden Dateien werden „Intermediate Files“ genannt und an den Xilinx-Kompiler weitergegeben. Danach findet die HDL-Kompilation, Analyse und Synthese statt, wobei der HDL-Code in digitale Logikelemente verwandelt wird. Der nächste Schritt trennt die Logik zwischen den physikalischen Bausteinen auf dem FPGA, was „Mapping“ genannt wird. Vor dem Generieren der Bit-Datei erfolgt im letzten Schritt des Kompiliervorgangs die physikalische Verteilung der Bausteine auf dem FPGA. Dadurch werden schließlich die Verbindungen zwischen den Logikblöcken hergestellt [48].

6.7.2 Kompilierung des FPGA-Codes

Der funktionsfähige Algorithmus für die I/Q-zu-ZF-Transformation wurde nun in den bereits vorhandenen Beispielcode des USRP-RIOs eingebunden.

Bei der Kompilierung des Algorithmus im Beispielcode ist ein „Timing Violation“ Fehler, in Abbildung 47 dargestellt, aufgetreten.

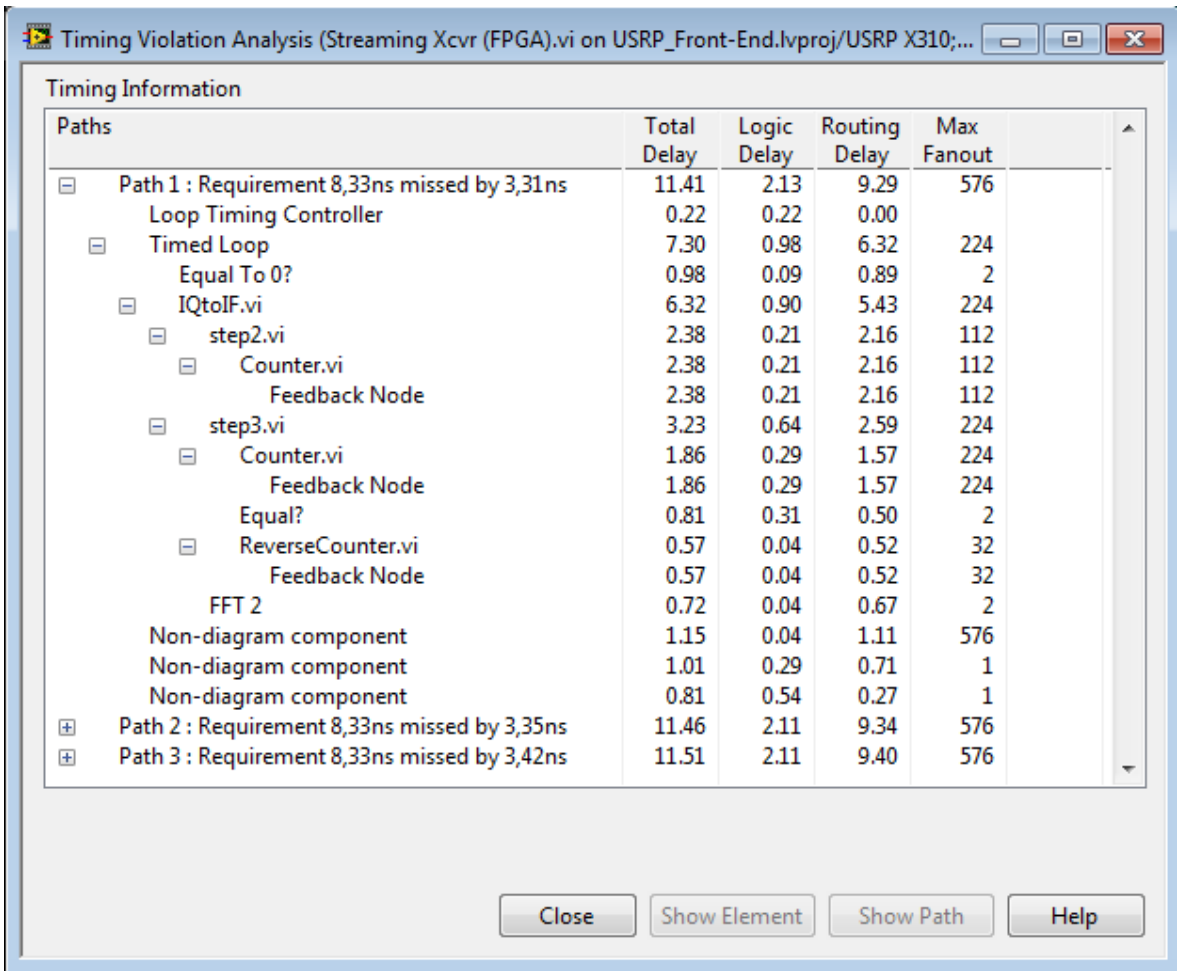


Abbildung 47: Timing Violation während des Kompiliervorgangs

Dies bedeutet, dass der FPGA-Code nicht innerhalb der erlaubten Zeit eines Takts ausführbar ist. Bei einem Takt von 120 MHz darf der Pfad eine Zeit von 8,333 ns nicht überschreiten:

$$\frac{1}{120\ MHz} = 8{,}333\ ns$$

Der Gesamtdurchlauf des Pfads benötigt bei der I/Q-zu-ZF-Transformation 11,41 ns, was einen Fehler von 3,31 ns nach sich zieht. Bei der Implementierung des Algorithmus im eigenen Projekt wurde mit einer Taktrate von 40 MHz gearbeitet. Der Takt des Beispielcodes beträgt 120 MHz. Dieser Takt ist abhängig von der maximalen Abtastrate des Transceivers von 120 MS/s und darf nicht geändert werden. Um dieses Problem zu beheben, ist es notwendig den Code zu optimieren oder in einer zusätzlichen Schleife mit niedrigerer Taktrate zu betreiben. Da der gesamte Verzug des Algorithmus 11,41 ns ergibt, wurden für den maximal erlaubten Takt der I/Q-zu-ZF-Transformation 87,642 MHz ermittelt:

$$\frac{1}{11{,}41\ ns} = 87{,}642\ MHz$$

Der verwendete Takt im FPGA wird von der internen Onboard-Clock oder dem Clock des Transceiver-Moduls bereitgestellt. Die Onboard-Clock des USRP-RIOs beträgt 40 MHz und

der des Transceiver-Moduls 120 MHz. Von diesen beiden Takten kann durch das Multiplizieren und Dividieren ein alternativer Takt abgeleitet werden.

Es wurde eine neue Schleife mit einen Takt von 60 MHz erzeugt, in welcher die beiden Algorithmen arbeiten sollen. Mittels eines FIFOs werden die Samples den verschiedenen Taktschleifen übergeben. Um Daten innerhalb von Schleifen mit verschiedenen Takten auf einem FPGA auszutauschen, können nur Target-Scoped FIFOs mit Implementierung als Block RAM verwendet werden. Andere Arten von FIFOs sind bei dieser Verwendungsweise unzulässig. Mit dieser Umsetzung und unter besserer Einhaltung der Zeitbedingungen war ein erfolgreiches Kompilieren möglich.

Nach erfolgreicher Kompilierung funktionierte der Code mit einer maximalen Abtastrate von 5 MS/s. Eine Abtastrate von 5 MS/s des Transceivers entspricht nach der I/Q-zu-ZF-Transformation 10 MS/s. Um die komplette GNSS-Bandbreite abzudecken, ist eine Bandbreite von 20 MHz notwendig. Der Grund, weshalb sich keine höhere Abtastrate erreichen lässt, liegt an dem Datendurchsatz des kompletten Algorithmus. Dieser ist abhängig von dem langsamsten Glied in der Punkt-zu-Punkt-Abarbeitung, welches die vorkommenden IFFT der IQ-zu-ZF-Transformation mit einem Datendurchsatz von 5,02 Zyklen/Sample darstellt. In Abhängigkeit der Taktrate und des Datendurchsatzes ergibt sich eine maximal erreichbare Abtastrate von 10 MS/s:

$$\frac{60\ MHz}{6\ Zyklen/Sample} = 10\ MS/s$$

Dieser Wert bezieht sich nicht auf die Abtastrate der Steuersoftware, sondern auf die durch die I/Q-zu-ZF-Transformation bedingte Verdopplung der Abtastrate. Die Eingangswerte können somit maximal mit der Hälfte des ermittelten Wertes abgetastet werden.

Um eine höhere Abtastrate zu erreichen, gibt es folgende Möglichkeiten:

- Die FFT und IFFT in die Hauptschleife mit 120 MHz Takt geben;
- den Code zu optimieren, um die Taktrate erhöhen zu können;
- den Datendurchsatz der FFT erhöhen.

Bei der ersten Möglichkeit müsste der Algorithmus der I/Q-zu-ZF-Transformation aufgeteilt werden. Das ist nicht gewünscht, weil der Algorithmus als komplettes VI nutzbar sein soll, um die Modularität aufrecht zu erhalten. Da Variante 2 einen zusätzlichen Aufwand bei der Implementierung darstellt, wurde Variante 3 gewählt. In dem Konfigurationsfenster der FFT/IFFT gibt es die Möglichkeit, den Datendurchsatz zu verändern. Standardmäßig ist immer der höhere Datendurchsatz eingestellt. Es ist möglich, diesen auf 1 Zyklus/Sample zu erhöhen, wodurch auch der Ressourcenverbrauch auf dem FPGA steigt.

Somit ergibt sich für die I/Q-zu-ZF-Transformation ein Gesamtdatendurchsatz von 2 Zyklen/Sample. Mit einer Taktfrequenz von 60 MHz der zusätzlichen Schleife ergibt sich ein maximal erreichbarer Datendurchsatz von 30 MS/s. Dieser ist ausreichend, um die gesamte GNSS-Bandbreite abzudecken:

$$\frac{60\ MHz}{2\ Zyklen/Sample} = 30\ MS/s$$

Nach den erforderlichen Anpassungen ergab sich bei der Kompilierung des Quellcodes folgender Verbrauch von Ressourcen auf dem FPGA, dargestellt in Abbildung 48. Es ist ersichtlich, dass auf dem verbauten FGPA des USRP-RIOs durchaus mehr und aufwändigere Algorithmen implementiert werden können.

Reports

Final device utilization (map)

Device Utilization	Used	Total	Percent
Total Slices	23016	63550	36,2
Slice Registers	63015	508400	12,4
Slice LUTs	57521	254200	22,6
DSP48s	508	1540	33,0
Block RAMs	178	795	22,4

Abbildung 48: Ressourcenverbrauch des FPGAs

Sind alle Bedingungen für einen reibungslosen Ablauf gegeben, kann mit dem Testlauf der GNSS-Störsignalunterdrückungseinheit begonnen werden.

6.8 Testlauf und Ergebnisse

Um die Funktionsweise der GNSS-Störsignalunterdrückungseinheit zu testen, wurden folgende Tests durchgeführt: Zuerst wurde ein Test unter Verwendung eines Front-Ends für den Softwareempfänger durchgeführt. Der zweite Test bezieht sich auf den implementierten Algorithmus zur Störsignalunterdrückung. Abschließend wurde die Repeater-Funktion kurz getestet.

6.8.1 Test als Front-End ohne Störsignalunterdrückung

Der Testlauf der GNSS-Störsignalunterdrückungseinheit als Front-End für den Softwareempfänger erfolgte mit realen GNSS-Signalen. Dazu wurde eine GNSS-Antenne über einen Bias Tee an das Transceiver-Modul des USRP-RIOs angeschlossen. Der FFT-Unterdrückungsalgorithmus wurde deaktiviert. Ein erfolgreiches Tracken der Satellitensignale im Softwareempfänger bestätigte die Funktionsweise der GNSS-Störsignalunterdrückungseinheit als Front-End.

Da der Softwareempfänger in Echtzeit nur mit dem eigenen Front-End NavPort-4 arbeitet, mussten die Samples des USRP-RIOs im „post-processing“ Modus übergeben werden. Dazu wurde eine bereitgestellte DLL-Datei in den Code des Host-Rechners eingebunden. Diese DLL-Datei bewirkt ein Übergeben der digitalen Signalfolge von dem Host-Rechner direkt an den darauf laufenden Softwareempfänger.

Im „post-processing“ Betrieb des Softwareempfängers müssen zwei Dateien zur Verfügung stehen: Das Sample (bei Verwendung mit dem USRP-RIO besser genannt der Stream), welches das digitalisierte GNSS-Signal enthält, und eine Meta-Datei mit Eigenschaften der

Samples für den Softwareempfänger. In der Meta-Datei mussten die Angaben über die Abtastrate und der Zwischenfrequenz an den Parametern des USRP-RIOs angepasst werden. Diese wurden wie folgt bestimmt:

$$ZF = \frac{BW}{2} + \frac{2 * BW}{output\ \frac{I}{Q}\ zu\ IF\ Transformation}$$

$$Abtastrate = 2 * BW$$

Da das USRP-RIO mit verschiedenen Abtastraten getestet wurde, erfolgte eine einmalige Berechnung der notwendigen Werte. Diese sind in Tabelle 20 notiert.

Tabelle 20: Werte für die Meta-Datei

Abtastrate USRP-RIO (BW)	**Abtastrate für Meta-Datei**	**ZF für Meta-Datei**
2 MS/s	4 MHz	1,015625 MHz
4 MS/s	8 MHz	2,031250 MHz
5,5 MS/s	11 MHz	2,79296875 MHz
10 MS/s	20 MHz	5,078125 MHz
10,24 MS/s	20,48 MHz	5,12000008 MHz

Bei den ersten Probeläufen hat sich herausgestellt, dass die intern verbaute Quarzuhr des USRP-RIOs in seiner Genauigkeit nicht ausreichend ist. Im Softwareempfänger wurde die Trägerphase der GNSS-Signale aufgrund der minderen Qualität des verbauten OCXO nicht erkannt [32]. Theoretisch ist ein temperaturkompensierter Quarzoszillator (engl.: Temperature Compensated Crystal Oscillator; Abk.: TCXO) ausreichend. Deshalb wurde für die weiteren Untersuchungen der GNSS-Störsignalunterdrückungseinheit eine externe Referenzquelle verwendet.

Trotz Einhaltung des Datendurchsatzes auf dem FPGA, wurde die notwendige Abtastrate des USRP-RIOs von 10 MS/s nicht erreicht. Es kam zu sogenannten Überläufen bei der DMA-Übertragung, dargestellt in Abbildung 49.

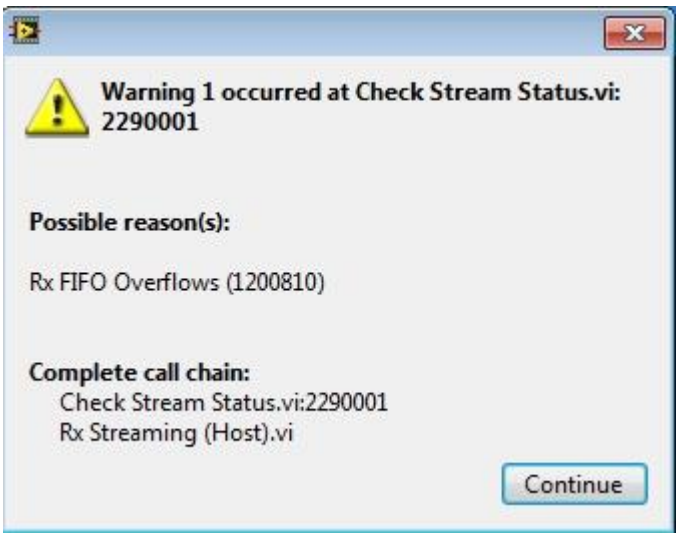


Abbildung 49: FIFO-Überlauf

Mit einer Abtastrate von 5 MS/s funktionierte die Datenübertragung. Für den Empfang von Galileo-Signalen ist jedoch eine höhere Bandbreite notwendig. Es müssen mindestens 10 MS/s erreicht werden. Dazu wurde die Abtastrate erhöht, wodurch sogenannte Überläufe bei der Datenübertragung der Samples vom FPGA auf den Host-Rechner auftraten. Diese Überläufe entstehen, wenn der Speicher schneller mit Daten beschrieben wird, als diese abgeholt werden. Um einen höheren Datendurchsatz zu erreichen, galt es den Engpass zu ermitteln und einzugrenzen. Folgende Punkte wurden betrachtet:

- Datenrate reduzieren;
- Abholen größerer Datenblöcke vom FPGA;
- Erhöhen der Schleifengeschwindigkeit beim Abholen auf dem Host-Rechner;
- Erhöhen der Speichertiefe auf FPGA oder/und Host-Rechner;
- Reduzieren der CPU-Last [44].

Das Reduzieren der Datenrate ist keine Option, weil das Ziel darin besteht, diese zu erhöhen.

Die Größe der abzuholenden Datenpakete ist anhand der Paketgröße für den Softwareempfänger festgelegt. Diese kann verringert oder vergrößert werden. Die Datenpakete müssen dazu auf dem Host-Rechner wieder in die richtige Größe gebracht werden. Es wurden keine Änderungen vorgenommen, um unnötige Rechenoperationen auf dem Host-Rechner zu vermeiden.

In LabVIEW kann innerhalb einer While-Schleife eine Wartefunktion mit beliebiger Zeitkonstante eingesetzt werden. Dies wird empfohlen, um die Last der CPU bei unendlich schneller Ausführung der Schleife zu schonen, indem die Schleife alle bestimmten Sekunden durchlaufen wird. In der Software der GNSS-Störsignalunterdrückungseinheit wird die Schleife bereits ohne Wartefunktion betrieben. Die Daten müssen, sobald sie bereitstehen im Speicher des FPGAs abgeholt werden.

Ein anderer Punkt befasst sich mit der Erhöhung des Speichers bei der DMA-Übertragung. Dieser wird auf dem FPGA und auf dem Host-Rechner allokiert. Dabei ist die Größe auf

dem FPGA standardmäßig mit 1023 Elementen vorgegeben. Anhand der LabVIEW-Anleitung ist dieser Wert für den Puffer für die meisten Anwendungen optimal und muss nicht geändert werden. Laut dem Bericht des Kompilierers ist noch Block RAM auf dem FPGA frei, welcher das Erhöhen erlauben würde. Davon wurde jedoch abgesehen. Einen interessanteren Aspekt stellt der Speicher auf dem Host-Rechner dar. Dieser ist standardmäßig mit 10.000 Elementen vorgegeben. Im Beispielprogramm des USRP-RIOs ist die Größe des allokierten RAM auf dem Host-Rechner von der Anzahl der abzuholenden Elemente multipliziert mit einen Faktor 8 definiert. Bei der verwendenden Anzahl an abzuholenden Elementen von 675.840 multipliziert mit dem Faktor 8 ergibt sich eine Speichertiefe von 5.406.720 Elementen. Ein Element besitzt eine Größe von 32 Bit, somit ergibt sich ein allokierter Wert von 21,63 MByte auf dem Host-Rechner. Angesichts der Tatsache, dass heutige Rechner über einen Arbeitsspeicher von mehreren Gigabyte verfügen, wurde dieser mit einer Tiefe von 100M Elementen erhöht, was zu einer Speicherzuweisung von 400 MByte führt.

Letzter Punkt befasst sich mit der Auslastung der CPU. Da Windows ein Multitasking-Betriebssystem ist, laufen im Hintergrund gleichzeitig eine Reihe von anderen Prozessen. Windows sollte sich vorrangig auf die Ausführung der GNSS-Störsignalunterdrückungseinheit und den darauf laufenden Softwareempfänger konzentrieren. Deswegen gilt es, alle nicht benötigten Programme und Anwendungen zu beenden. In den Windows Systemeinstellungen sollte deshalb unter „Energieoptionen“ der Energiesparplan auf Höchstleistung gestellt werden.

Unter Einhaltung der oben angegeben Methoden zur Minimierung von FIFO-Überläufen war nur ein minimaler Erfolg gegeben. Es kam sporadisch immer wieder zu sogenannten „Überläufen“, weshalb nach einer weiteren Lösung des Problems gesucht werden musste.

Als Datenübertragung vom FPGA auf dem Host-Rechner dient im verwendeten Beispielprogramm ein DMA-FIFO mit einem 32 Bit Integer-Datentyp. Dieser 32 Bit Integer setzt sich aus den zwei Datenkanälen auf dem FPGA zusammen. Diese sind der I- und Q-Kanal mit einer Größe von jeweils 16 Bit. Nach der I/Q-zu-ZF-Transformation wird nur noch ein Kanal genutzt, was zu einer ineffizienten Ausnutzung des FIFOs bei der Datenübertragung auf den Host-Rechner führt. Deshalb gilt es, den verwendeten Pfad mittels eines Splitters auf dem FPGA in zwei Pfade aufzuteilen und parallel an den Host-Rechner zu übertagen. Sind die Samples an dem Host-Rechner angekommen, werden diese wieder zusammengefügt und dem Softwareempfänger übergeben. Dadurch erhöht sich der Datendurchsatz bei der Übertragung um den Faktor 2.

Nach Umsetzen des Splitters war es möglich, die GNSS-Störsignalunterdrückungseinheit mit der erforderlichen Abtastrate von 10 MS/s zu betreiben.

6.8.2 Test als Front-End mit Störsignalunterdrückung

Wie sich die GNSS-Störsignalunterdrückungseinheit im Vergleich zu einem kommerziellen Front-End bei einer auftretenden Störungen der GNSS-Signale verhält, wird in folgendem Versuch gezeigt. Die GNSS-Signale wurden mit dem GNSS-Signalsimulator mit den

gleichen Parametern der Sendeleistung wie in Kapitel 4.3.1 erzeugt. Das Störsignal wurde über ein programmierbares Dämpfglied mit dem GNSS-Signal kombiniert. Die Leistung der Störung wurde im zeitlichen Verlauf mit einer Rampenfunktion alle 10 s erhöht. Am Ende erfolgte die Bereitstellung des störbehafteten GNSS-Signals auf dem kommerziellen Front-End NavPort-4 und dem USRP-RIO über einen Splitter. Dabei wurden identische Störungen wie in Kapitel 5.3 betrachtet. Die Auswertung erfolgte in den folgenden Plots mit den beiden im Softwareempfänger erzeugten RINEX-Dateien. Betrachtet wurde der Galileo-Satellit Nummer 24.

Bevor mit den Versuchen gestartet wurde, mussten beide Front-Ends dem gleichen Leistungslevel angepasst werden, um eine identische Signalqualität zu erhalten. Dazu wurden der AGC-Wert des Front-Ends und der digitale Verstärkungsfaktor des USRP-RIOs angepasst. Die Amplitude bei beiden weist einen Wert von 10 Digits auf und die Leistung lag bei -60 dB/MHz. Die Überprüfung erfolgte mit den im Softwareempfänger angezeigten Spektren. Das Träger-zu-Rausch-Verhältnis bei den Galileo-Signalen betrug bei beiden Softwareempfängern ca. 45 dB-Hz. Der AGC-Wert vom NavPort-4 wurde mit 30 festgelegt. Der digitale Verstärkungsfaktor des USRP-RIOs mit 0,19. Der optimal ermittelte Schwellwert des FFT-Unterdrückungsalgorithmus wurde mit einem Faktor von 20 bestimmt.

Zunächst wurde das NavPort-4 und das USRP-RIO mit der ersten Störung, den vier CW, untersucht. Abbildung 50 zeigt das Ergebnis der ausgewerteten RINEX-Dateien beider Empfänger.

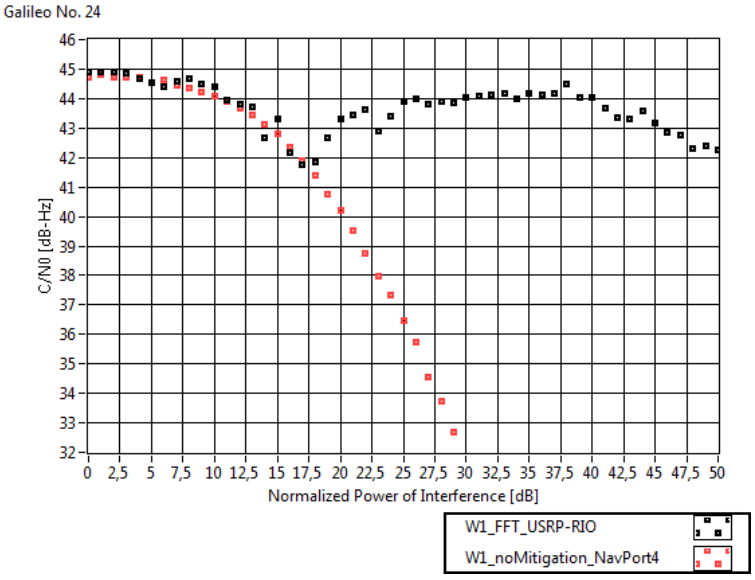


Abbildung 50: Ergebnis vier CW mit NavPort-4 und USRP-RIO

Vier kontinuierliche Schwingungen als Störung müssen mit dem FFT-Unterdrückungsalgorithmus ein besseres Ergebnis liefern, was aus Kapitel 5.5 ersichtlich ist. Um eine höhere Signaldynamik zu erreichen, muss eine höhere Leistung des GNSS-Signals vorliegen oder die interne Rechengenauigkeit im FPGA erhöht werden. Da mit dem GNSS-Signalsimulator keine höhere Leistung wählbar ist, wurde am Ausgang des GNSS-Simulators ein HF-Verstärker geschalten, welcher die Leistung der GNSS-Signale um 15

dB erhöht. Durch mehr Leistung resultiert eine höhere FXP-Auflösung im FPGA, was eine feinere Schwellwertbestimmung zuließ. Durch die höhere Leistung der GNSS-Signale mussten die Parameter beider Front-Ends (NavPort-4 und USRP-RIO) neu ermittelt werden. Es ergab sich beim NavPort-4 ein neuer AGC-Wert von 60. Der digitale Verstärkungsfaktor beim USRP-RIO beträgt 0,04.

Nach der Justierung der Werte wurden alle vier Versuche durchlaufen. Es wurden folgende Störungen mit der GNSS-Störsignalunterdrückungseinheit betrachtet: vier CW, AWGN, Radar und ein Chirp-Signal. Die Ergebnisse sind in den nachfolgenden Grafiken dargestellt. Grün stellt das NavPort-4 ohne Unterdrückung der Störung dar und rot das USRP-RIO mit dem FFT-Unterdrückungsalgorithmus. Mit dem Softwareempfänger des NavPort-4 Front-Ends wurde das Signal aufgezeichnet und mit dem den CPU-gestützten FFT-Algorithmus mit einer FFT-Länge von 1024 prozessiert. Dieser ist in schwarz dargestellt.

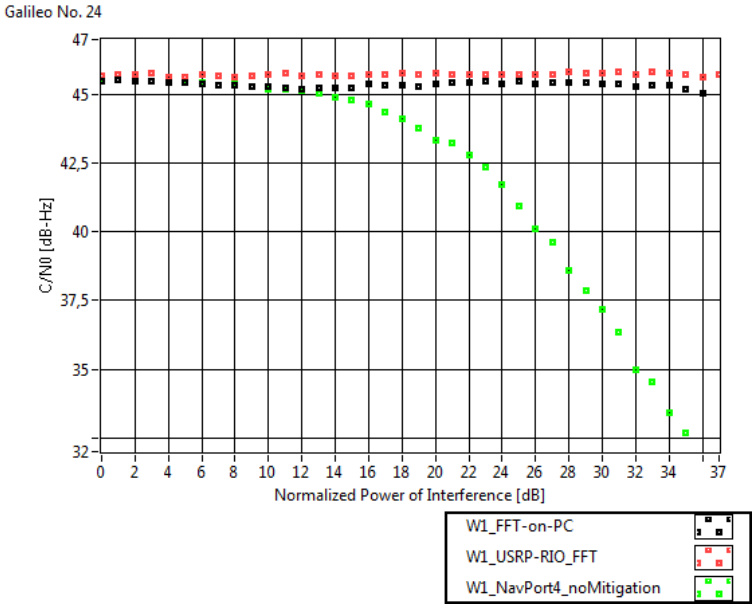


Abbildung 51: Ergebnis der Störsignalunterdrückung mit FPGA - vier CW

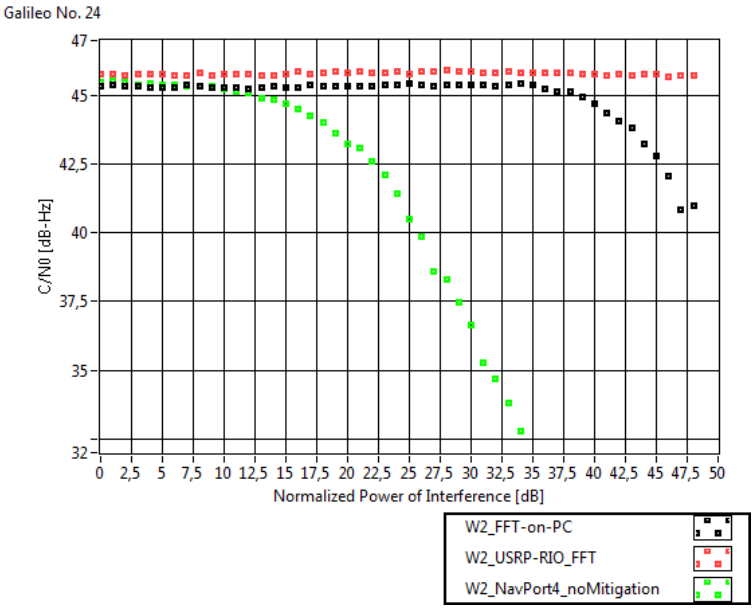


Abbildung 52: Ergebnis der Störsignalunterdrückung mit FPGA - AWGN

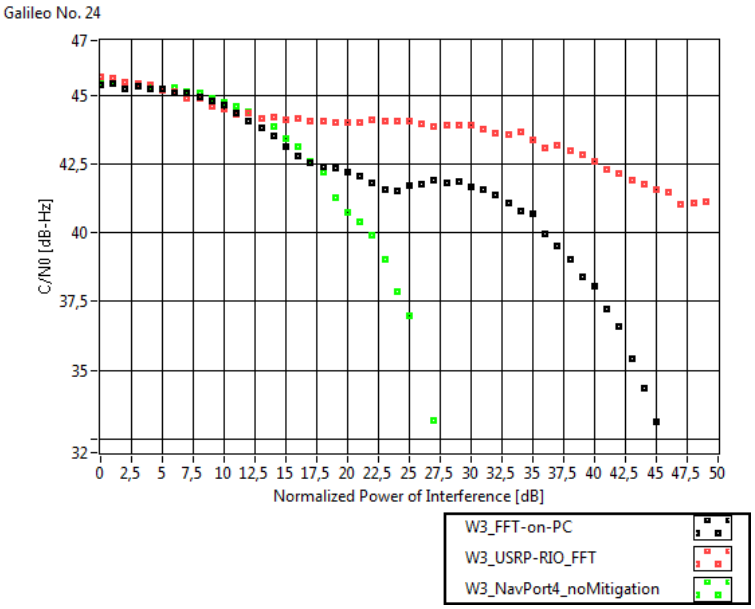


Abbildung 53: Ergebnis der Störsignalunterdrückung mit FPGA – Chirp-Signal

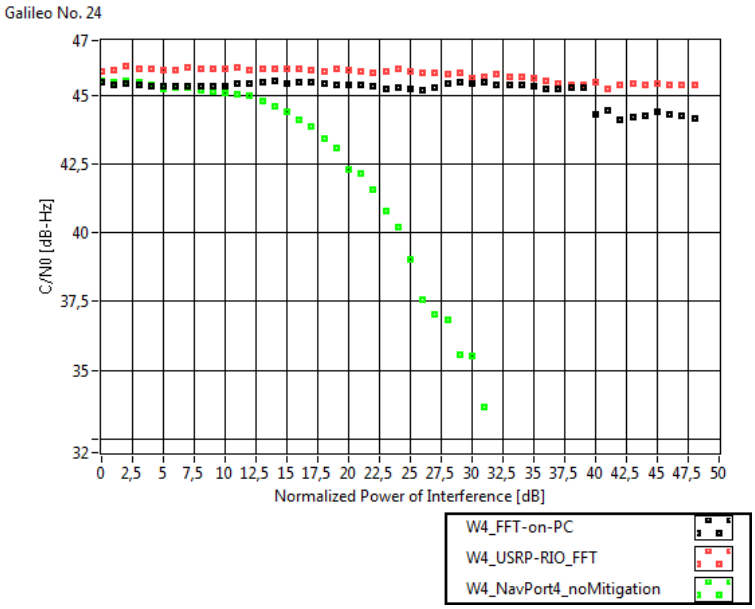


Abbildung 54: Ergebnis der Störsignalunterdrückung mit FPGA - Radar

Der FFT-Unterdrückungsalgorithmus auf dem FPGA funktioniert mit allen in Betracht gezogenen Störsignalen. Teilweise liefert die Implementierung dieses Algorithmus zur Störsignalunterdrückung ein besseres Ergebnis als der vergleichbare CPU-gestützte Algorithmus, trotz der reduzierten Auflösung im FPGA und mit der Verwendung des Festkommazahlenformates. Das USRP-RIO stellt im gesamten ein besseres Endsystem als das konventionell verwendete Front-End dar. In Abbildung 52 ist zu erkennen, dass mit dem CPU-gestützten Algorithmus ab einer normalisierten Leistung der Störung von 37,5 dB, eine Minderung des C/N_0 beginnt. Auf dem FPGA ist das C/N_0 bis zum Ende konstant. Bei dem konventionellen Front-End mit einer Quantisierung von 8 Bit tritt bei einer hohen Leistung eine Übersteuerung des ADC auf. Der ADC des USRP-RIOs arbeitet mit einer Quantisierung von 14 Bit.

6.8.3 Test als Repeater

Die Tests als Front-End mit und ohne Störsignalunterdrückung zeigten bereits Erkenntnisse über die richtige Arbeitsweise der beiden Algorithmen. Hierbei wurde sich nur kurz auf das Testen der Repeater-Funktion bezogen. Über eine GNSS-Antenne wurden Satellitensignale empfangen und an einen weiteren Empfänger übergeben. Der GNSS-Empfänger arbeitete einwandfrei, womit eine korrekte Arbeitsweise gegeben ist. Durch Zuschalten des FFT-Unterdrückungsalgorithmus sind identische Ergebnisse zu erwarten.

7 Schlusswort

Die Störsignalunterdrückung im Bereich von GNSS spielt in der heutigen Zeit und in der Zukunft aufgrund zunehmender Anzahl von Satellitennavigationsanwendung eine immer wichtigere Rolle. Es hat sich gezeigt, dass mit speziellen Algorithmen störbehaftete GNSS-Signale von ihren Störkomponenten befreit werden können. Damit ist trotz auftretender Störungen weiterhin eine Positionsbestimmung möglich. Jedoch ist bei den untersuchten CPU-gestützten Algorithmen meist keine Echtzeitfähigkeit gegeben. Deshalb wurde sich auf alternative Methoden zur digitalen Signalverarbeitung auf einem FPGA bezogen.

Mit der Umsetzung der GNSS-Störsignalunterdrückungseinheit auf einem SDR-System wurde deutlich, dass sich ein echtzeitfähiges Front-End oder ein Repeater mit Störsignalunterdrückung für eine große Anzahl an auftretenden Störsignalen auf einem FPGA realisieren lässt.

Auf den heutigen FPGAs stehen sehr viele Ressourcen zur Verfügung. Um den steigenden Anforderungen gerecht zu werden, können darauf weitere Algorithmen zur Störsignalunterdrückung implementiert werden. Eine gute Störsignalerkennung könnte den geeignetsten Algorithmus mit besten Paramatern automatisch auswählen.

Da das Erkennen von Störsignalen keinen rechenintensiven Aufwand darstellt, kann dieser Teil der Störsignalunterdrückung auf den Host-Rechner ausgelagert werden, was eine einfachere Implementierung ermöglicht.

Literaturverzeichnis

[1] Wikipedia: Globales Navigationssatellitensystem. URL: http://de.wikipedia.org/wiki/Globales_Navigationssatellitensystem, Abruf: 22.01.2014

[2] Hofmann-Wellenhof, Bernhard ; Lichtenegger, Herbert ; Walse, Elmar: GNSS – Global Navigation Satellite Systems : GPS, GLONASS, Galileo and more. – Wien : Springer, 2008

[3] U.S. Department of Homeland Security: GPS Constellation Status. URL: http://www.navcen.uscg.gov/?Do=constellationStatus, Abruf: 22.01.2015

[4] USA ; CE: Agreement on the Promotion, Provision and use of Galileo and GPS Satellite-based Navigation Systems and Related Applications. URL: http://ec.europa.eu/enterprise/policies/satnav/galileo/files/2004_06_21_eu_us_agreement_en.pdf, Abruf: 24.01.2015

[5] Wikipedia: GLONASS. URL: http://de.wikipedia.org/wiki/GLONASS, Abruf: 07.04.2015

[6] Hessin, Robert: U.S. Space-Based Positioning, Navigation and Timing Policy and Program Update. URL: http://www.unoosa.org/pdf/icg/2009/icg-4/02.pdf, Abruf: 24.01.2015

[7] Borre, Kai ; Akos, Dennis ; Bertelsen, Nicolaj ; Rinder, Peter ; Jensen, Soren Holdt: A Software-Defined GPS and Galileo Receiver : A Single-Frequency Approach. – Boston : Birkhäuser, 2007

[8] Dodel, Hans ; Häupler, Dieter: Satellitennavigation. – 2., korrigierte und erweiterte Aufl. – Heidelberg : Springer, 2010

[9] NAVILOCK®: Navilock NL-602U USB 2.0 GPS / Galileo Empfänger u-blox. URL: http://www.navilock.de/produkte/F_778_GPS_61840/merkmale.html, Abruf: 30.01.2015

[10] Thiel, Andreas ; Ammann Michael: Anti-Jamming techniques in u-blox GPS receivers. URL: https://www.u-blox.com/images/downloads/Product_Docs/u-blox%20anti-jamming_whitepaper_%28GPS-X-09008%29.pdf, Abruf: 30.01.2015

[11] NAVILOCK®: Navilock NL-442U USB 2.0 GPS Empfänger SiRFstarIV™. URL: http://www.navilock.de/produkte/F_978_Sirf-Star-IV_61994/merkmale.html, Abruf: 30.01.2015

[12] CSR: SiRFstarIV™ GSD4e. URL: http://www.csr.com/products/35/sirfstariv-gsd4e, Abruf: 30.01.2015

[13] Septentrio: PolaRx4TR PRO. URL: http://www.septentrio.com/sites/default/files/PolaRx4_DS_V112014_24%20web.pdf, Abruf: 30.01.2015

[14] EMVG (idF v. 26.02.2008) §6 Abs. I Satz 1

[15] TKG (idF v. 22.06.2004) $90

[16] Wikipedia: Radar. URL: http://de.wikipedia.org/wiki/Radar, Abruf: 04.04.2015

[17] Bauernfeind, Roland ; Krämer, Isabelle ; Beckmann, Hanno ; Eissfeller, Bernd ; Vierroth, Volker: In-Car Jammer Interference Detection in Automotive GNSS Receivers and Localization by Means of Vehicular Communication. Wien : IEEE, 2011

[18] IFEN GmbH: NavX-NCS User Manual. Poing, 2014

[19] National Instruments: NI USRP-2920 SDR-Transceiver mit 50 MHz bis 2,2 GHz. URL: http://sine.ni.com/nips/cds/view/p/lang/de/nid/212995, Abruf: 22.01.2015

[20] National Instruments: NI PXI-5695 Programmierbarer 8-GHz-RF-Dämpfer. URL: http://sine.ni.com/nips/cds/view/p/lang/de/nid/207936, Abruf: 06.02.2015

[21] National Instruments: NI PXIe-5665 Hochleistungs-Vektorsignalanalysator bis 14 GHz. URL: http://sine.ni.com/nips/cds/view/p/lang/de/nid/209379, Abruf: 06.02.2015

[22] IFEN GmbH: SX-NSR User Manual. Poing, 2014

[23] Gurtner, Werner: RINEX The Receiver Independent Exchange Format. URL: https://igscb.jpl.nasa.gov/igscb/data/format/rinex302.pdf, Abruf: 08.02.2015

[24] Köhne, Anja ; Wößner, Michael: NMEA-0183 Daten. URL: http://www.kowoma.de/gps/zusatzerklaerungen/NMEA.htm, Abruf: 08.02.2015

[25] Bauernfeind, Roland: Analyse, Detektion und Unterdrückung von InCar-GNSS-Störsendern in intelligenten Transportsystemen (InCarITS) : Abschlussbericht „InCar ITS“, Neubiberg, 2013, Förderkennzeichen: 50NA1001

[26] Capozza, Paul ; Holland, Brian ; Hopkinson, Thomas ; Landrau, Roberto: A Single-Chip Norrow-Band Frequency-Domain Excisor for a Global Positioning System (GPS) Receiver. In: IEEE Journal of Solid-State Circuits (2000), Nr. 3, S. 401-411.

[27] Kraus, Thomas ; Bauernfeind, Roland ; Sicramaz-Ayaz, Ayse: TERMINATE – Novel DSP Techniques and RFFE Design Selection. München, 2013

[28] Ernou, Y. ; Renard, A. ; Kirby E.: ATC Radar Interference Impact On Air Receiver. Portland : ION GNSS, 2003

[29] Kammeyer, Karl-Dirk ; Kroschel, Kristian: Digitale Signalverarbeitung : Filterung und Spektralanalyse mit MATLAB-Übungen. – 7., erweiterte und korrigierte Aufl. Wiesbaden : Vieweg+Teubner, 2009

[30] National Instruments: Windowing Optimizing FFTs Using Window Functions. URL: http://www.ni.com/white-paper/4844/en/, Abruf: 01.04.2015

[31] Tektronix: Understand FFT Overlap Processing. URL: https://cas.web.cern.ch/cas/Denmark-2010/Caspers/Tektronix%20%20primer%20on%20overlapping%20FFT%20signals%202009%20CAS2010.pdf, Abruf: 01.04.2015

[32] Pany, Thomas: Navigation Signal Processing for GNSS Software Receivers. – Norwood : ARTECH HOUSE, 2010

[33] National Instruments: NI USRP-2952R : 400 MHz to 4.4 GHz Tunable RF Transceiver. URL: http://www.ni.com/pdf/manuals/374412a.pdf, Abruf: 02.03.2015

[34] Wikipedia: Software Defined Radio. URL: http://de.wikipedia.org/wiki/Software_Defined_Radio, Abruf: 02.03.2015

[35] Xilinx Inc.: Kintex-7 FPGA Family. URL: http://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf, Abruf: 05.04.2015

[36] Kesel, Frank ; Bartholomä, Ruben: Entwurf von digitalen Schaltungen und Systemen mit HDLs und FPGAs. – München : Oldenbourg, 2006

[37] Xilinx Inc.: LogiCORE IP Fast Fourier Transform v8.0. URL: http://www.xilinx.com/support/documentation/ip_documentation/ds808_xfft.pdf, Abruf: 22.03.2015

[38] Xilinx Inc.: LogiCORE IP LTE Fast Fourier Transform v1.0. URL: http://www.xilinx.com/support/documentation/ip_documentation/lte_fft_xmp125.pdf, Abruf: 22.03.2015

[39] Wikipedia: Festkommazahl. URL: http://de.wikipedia.org/wiki/Festkommazahl, Abruf: 19.04.2015

[40] National Instruments: Using the Fixed-Point Data Type. URL: http://zone.ni.com/reference/en-XX/help/371599C-01/lvfpgaconcepts/fpgafixed-point/, Abruf: 19.04.2015

[41] National Instruments: How DMA Transfers Work. URL: http://zone.ni.com/reference/en-XX/help/371599H-01/lvfpgaconcepts/fpga_dma_how_it_works/, Abruf: 23.03.2015

[42] Kraus, Thomas: LabVIEW Projekt „IQtoIF“ : source-code Version 1.0. München, 2014

[43] National Instruments: LabVIEW FPGA Compilation Process : From Run Button to Bitfile. URL: http://www.ni.com/white-paper/9381/en/, Abruf: 10.04.2015

[44] National Instruments: Avoiding Buffer Errors in DMA Applications (FPGA Module). URL: http://zone.ni.com/reference/en-XX/help/371599G-01/lvfpgaconcepts/fpga_dma_fifo_buffer_size/, Abruf: 23.03.2015

[45] Kehtarnavaz, Nasser ; Mahotra, Sidharth: Digital Signal Processing Laboratory : LabVIEW-Based FPGA Implementation. Boca Raton : Brown Walker Press, 2010

[46] Wikipedia: Radio Technical Commission for Maritime Services. URL: http://de.wikipedia.org/wiki/Radio_Technical_Commission_for_Maritime_Services, Abruf: 31.03.2015

[47] Polikar, Robi: The Wavelet Tutorial. URL: http://users.rowan.edu/~polikar/WAVELETS/WTtutorial.html, Abruf: 09.03.2015

[48] National Instruments: Understanding the LabVIEW FPGA Compile System (FPGA Module). URL: http://zone.ni.com/reference/en-XX/help/371599H-01/lvfpgaconcepts/compiling_fpga_vis/, Abruf: 05.05.2015

[49] Universität der Bundeswehr München ; ESA/ESTEC: Interference Mitigation based on Novel Signal Processing Cancellation and RF-Front-End (TERMINATE). Vertragsnummer: 4000106305

[50] Kraus, Thomas ; Bauernfeind, Roland ; Eissfeller, Bernd: Survey of In-Car Jammers – Analysis and Modeling oft he RF Signals and IF Samples (Suitable for Active Signal Cancelation. Portland : ION GNSS, 2011

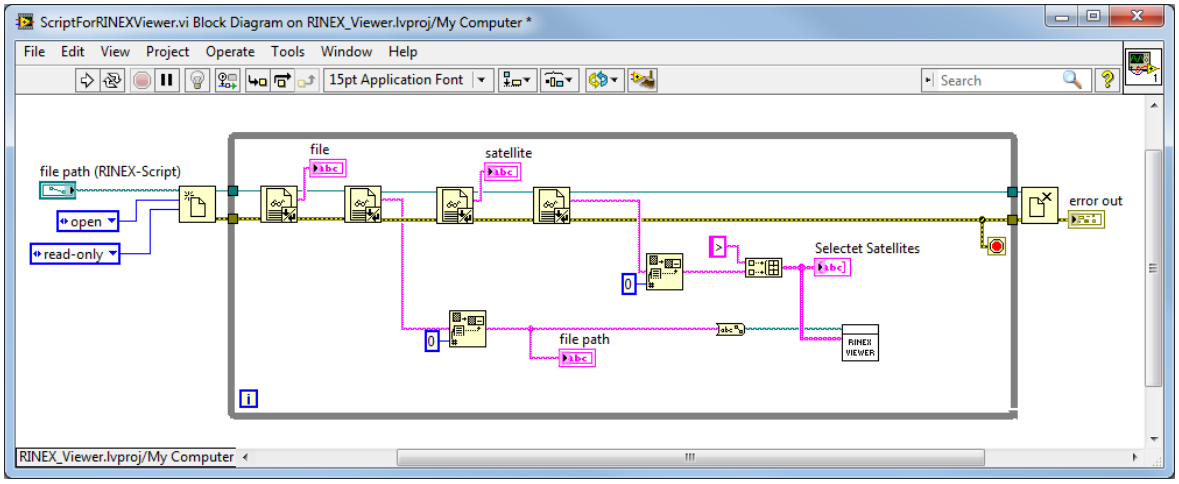
[51] Dovis, Fabio: GNSS Interference Threats and Countermeasures. Boston : Artech House, 2015

[52] Parkinson W., Bradford ; Spilker Jr., James J.: Global Positioning System: Theory and Applications Valume I. Washington, D.C. : American Institute of Aeronautics and Astronautics, 1996

Anlagen

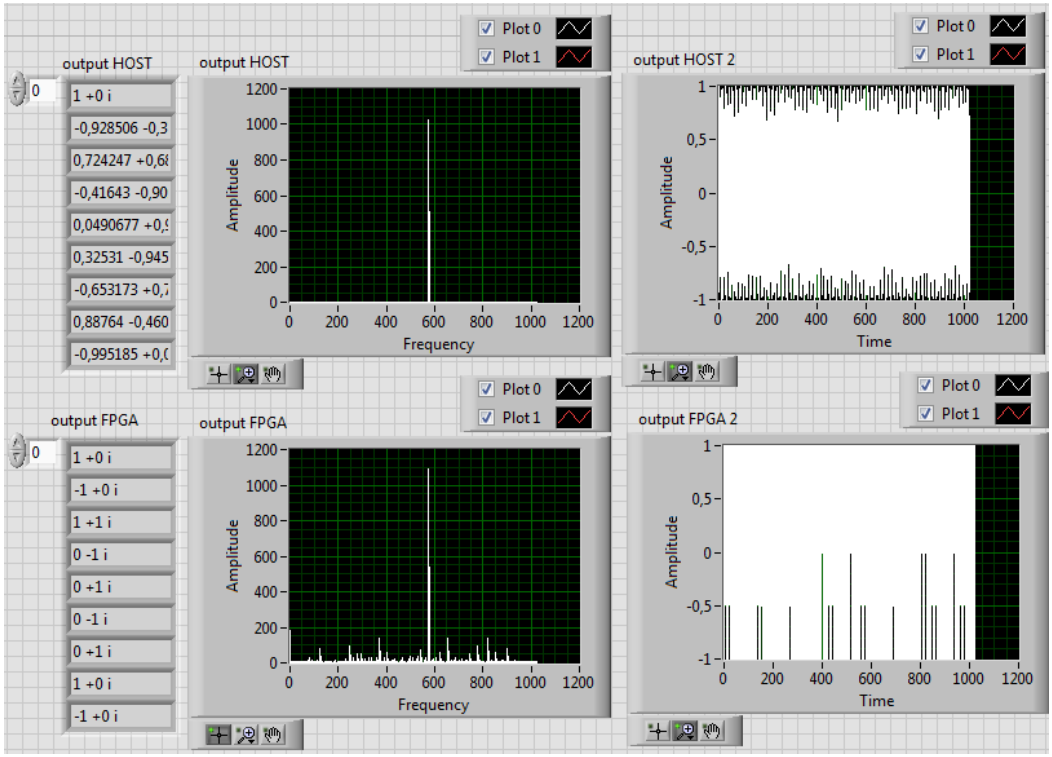
Anlagen, Teil 1

Quellcode des RINEX-Viewer für das automatisch Einlesen von mehreren RINEX-Dateien aus Kapitel 4.4.2.

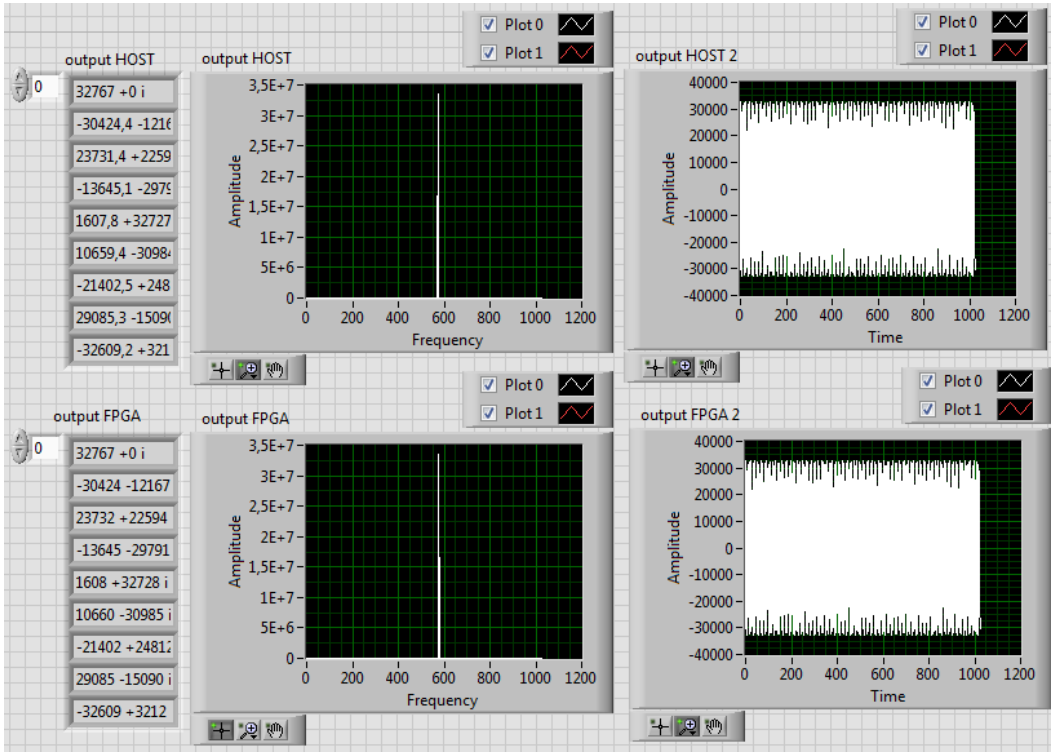


Anlagen, Teil 2

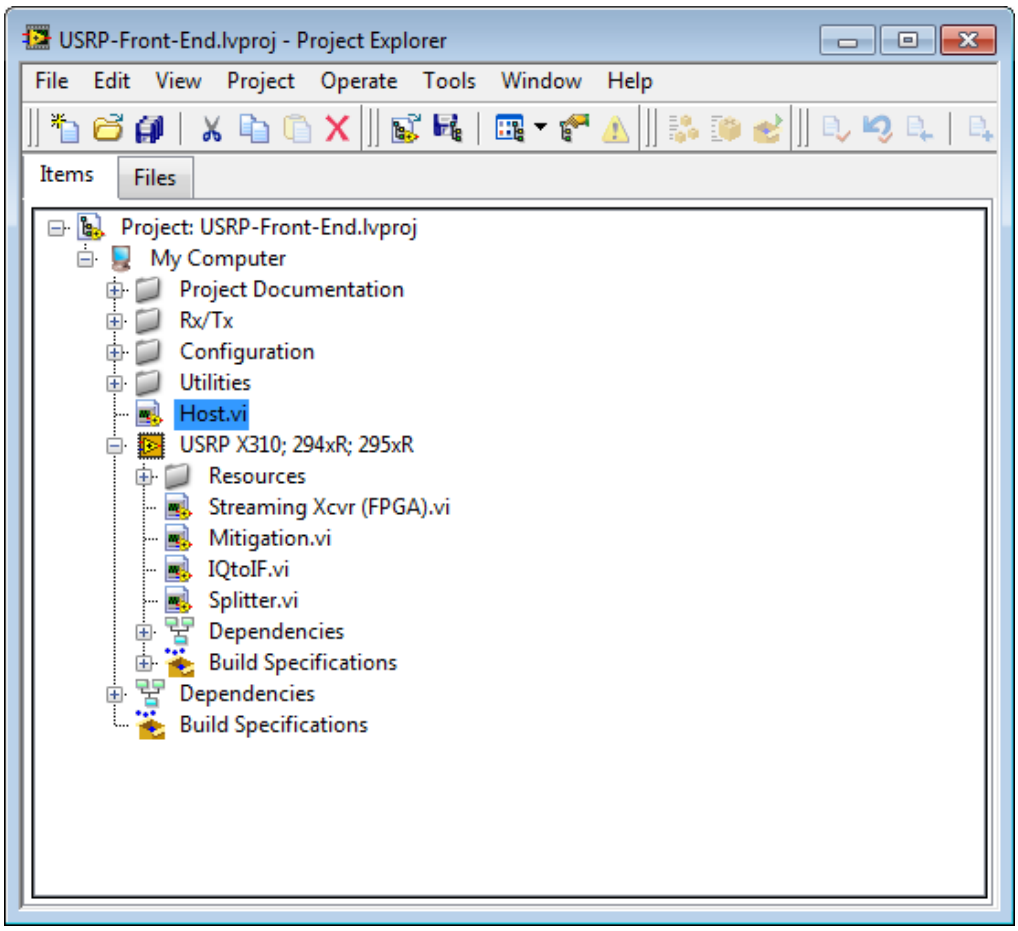
Die folgende Darstellung zeigt die Überprüfung der FXP-Reduzierung mit kleinstmöglicher Amplitude von 1 aus Kapitel 6.5.1.



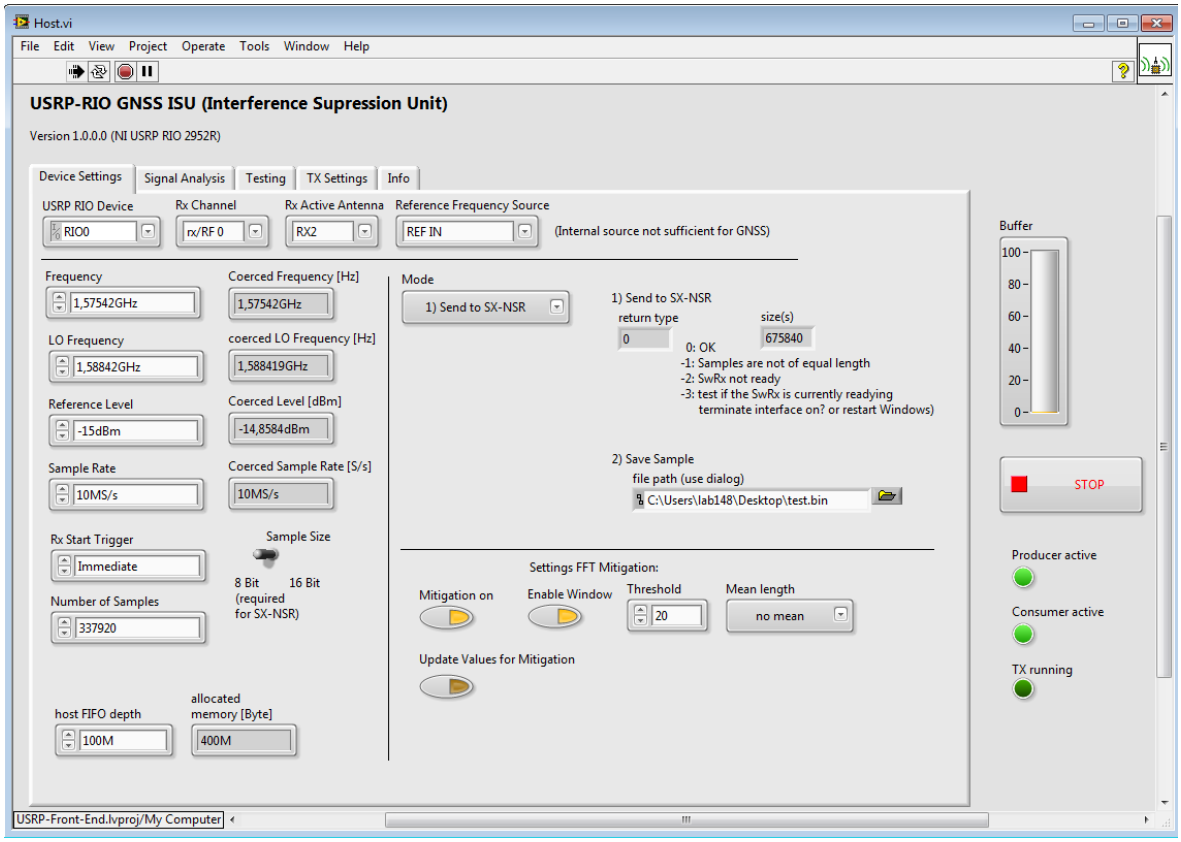
Die folgende Darstellung zeigt die Überprüfung der FXP-Reduzierung mit größtmöglicher Amplitude von 32767 aus Kapitel 6.5.1.



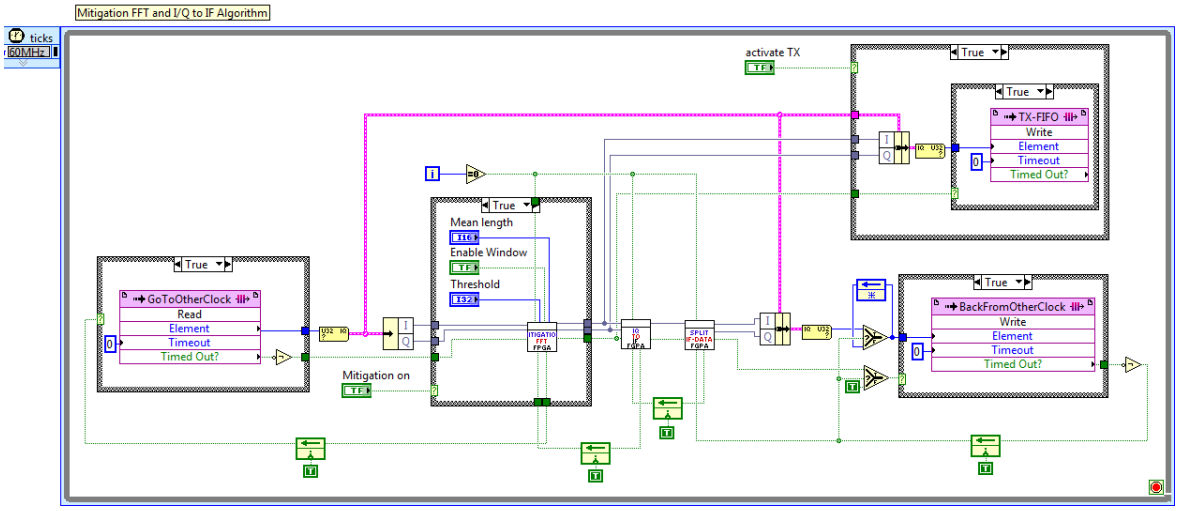
Projekt-Explorer der GNSS-Störsignalunterdrückungseinheit



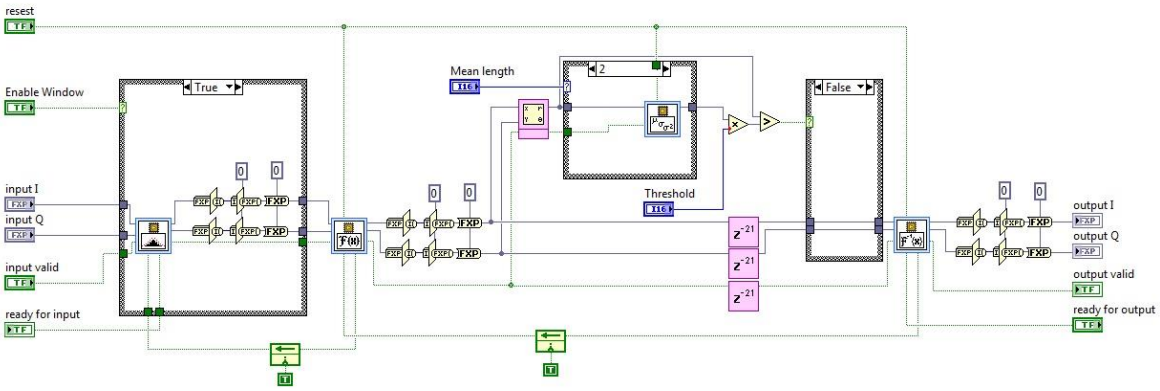
Steuersoftware der GNSS-Störsignalunterdrückungseinheit



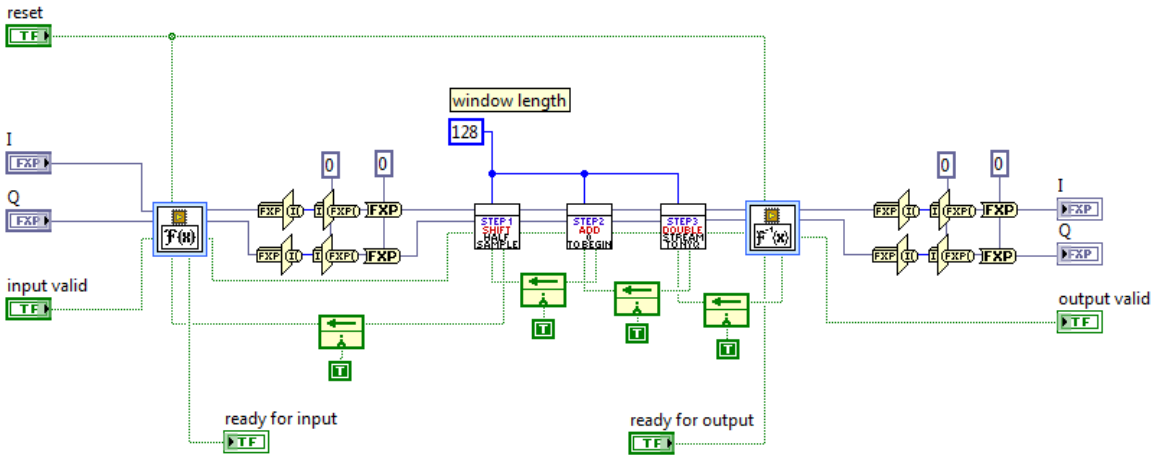
LabVIEW FPGA-Quellcode der Prozessierungsschleife mit FFT-Unterdrückungsalgorithmus, I/Q-zu-ZF-Transformation und Splitter



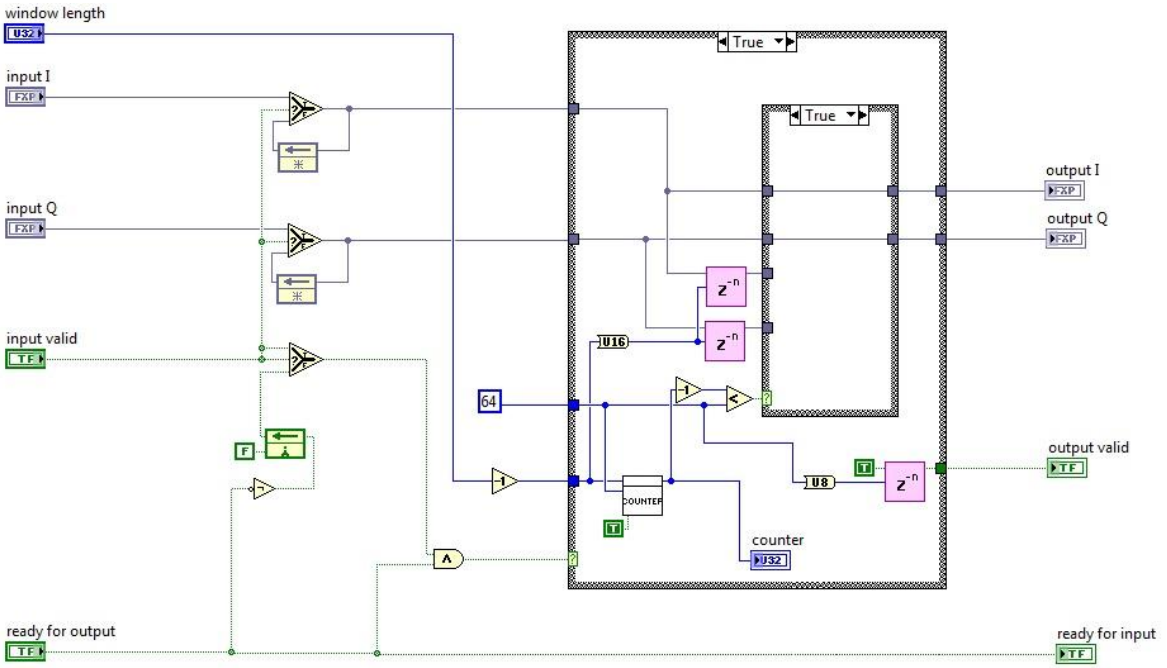
LabVIEW FPGA-Quellcode des FFT-Unterdrückungsalgorithmus



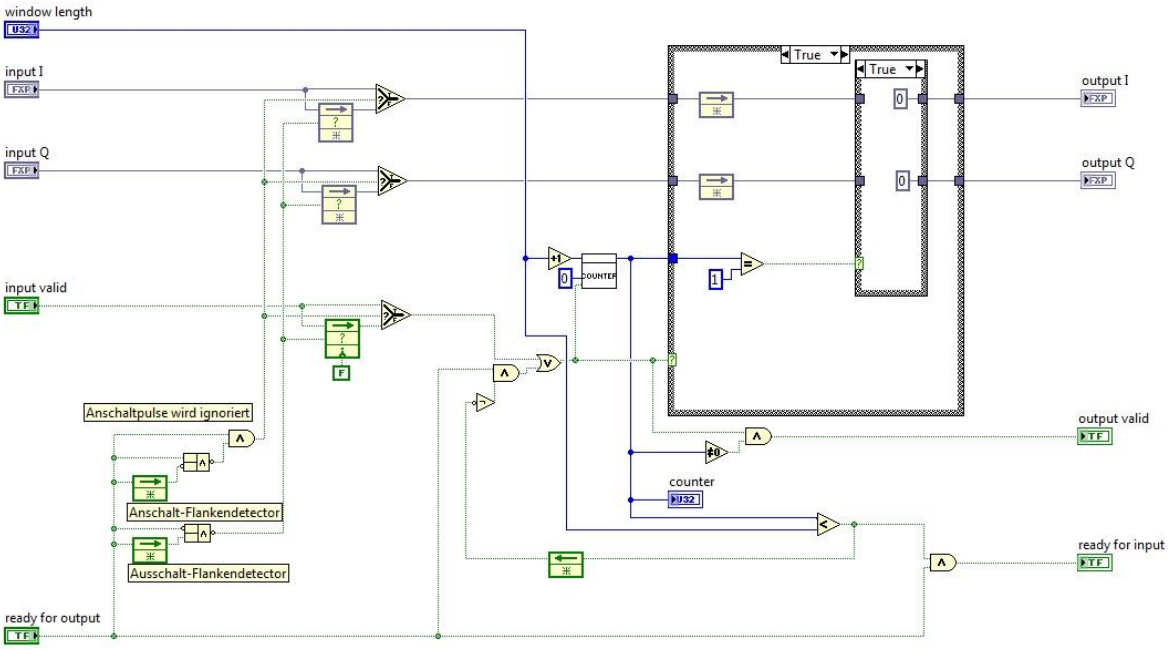
LabVIEW FPGA-Quellcode der I/Q-zu-ZF-Transformation



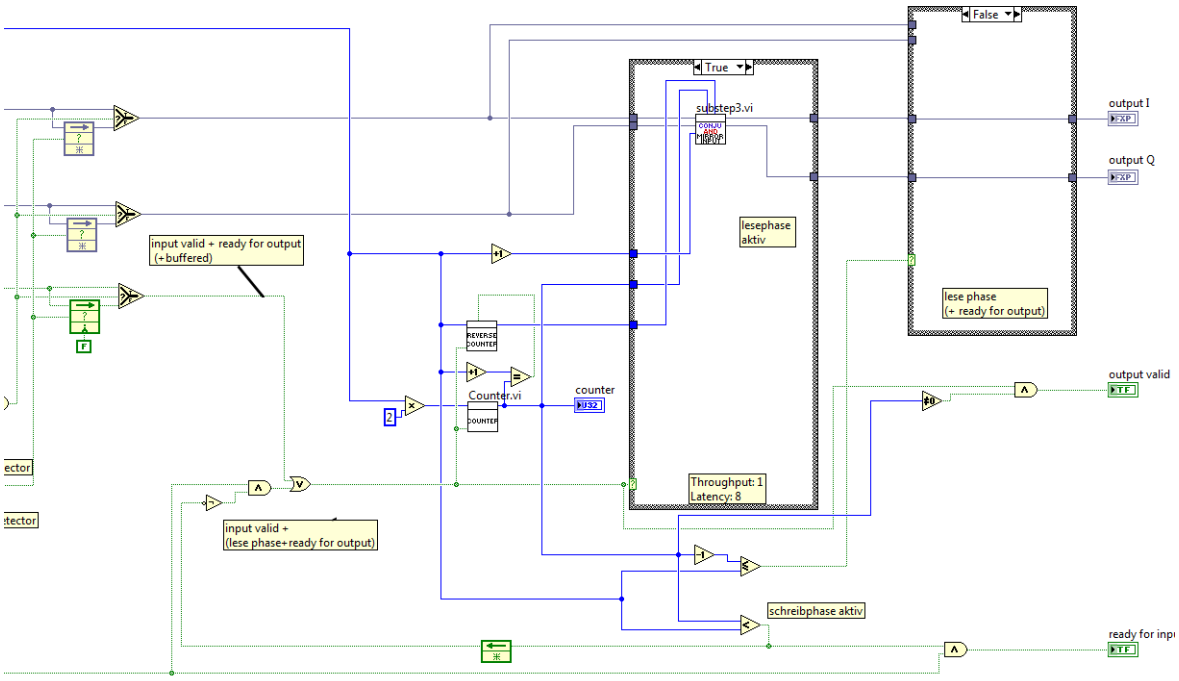
LabVIEW FPGA-Quellcode des Schritt 1 der I/Q-zu-ZF-Transformation



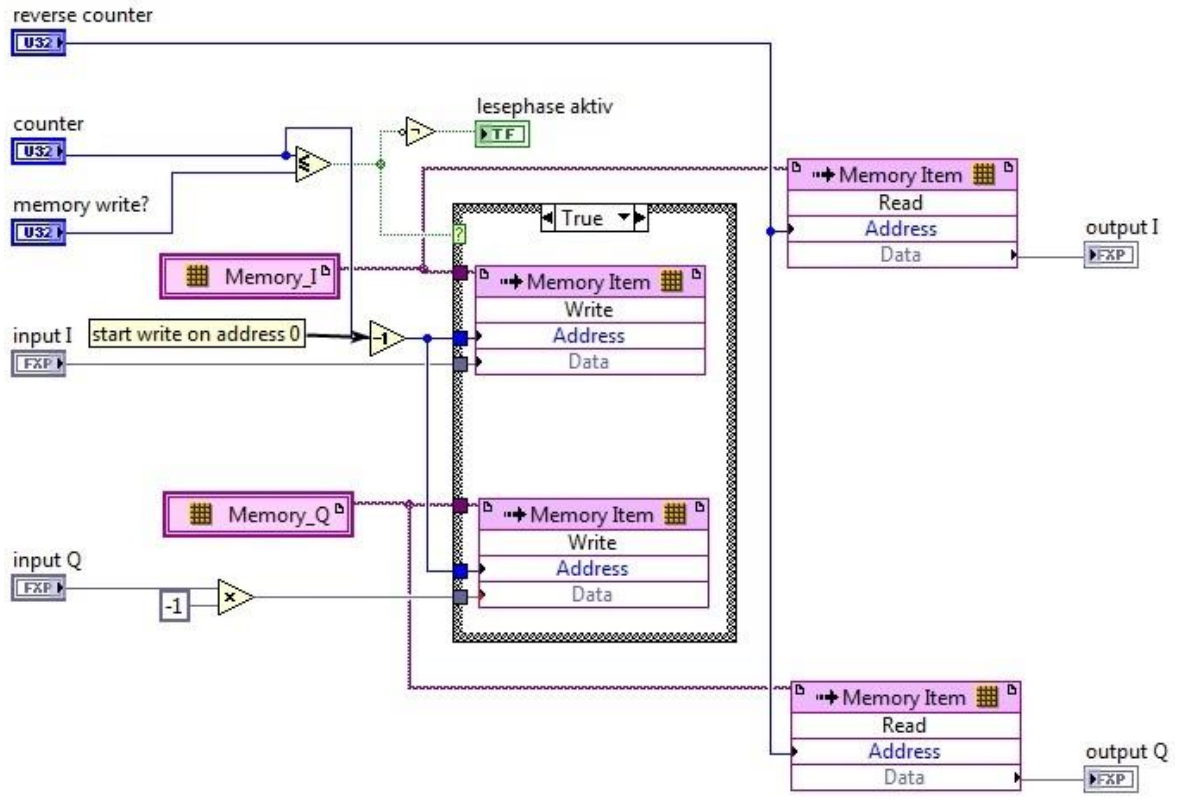
LabVIEW FPGA-Quellcode des Schritt 2 der I/Q-zu-ZF-Transformation



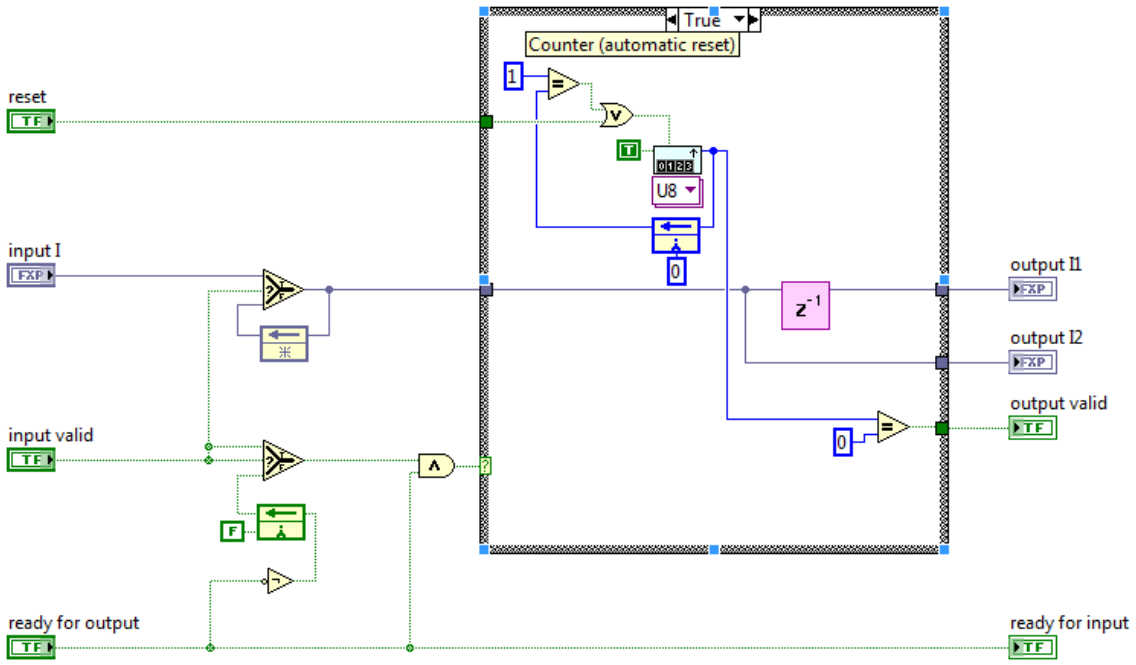
LabVIEW FPGA-Quellcode des Schritt 3 der I/Q-zu-ZF-Transformation



LabVIEW FPGA-Quellcode des subSchritt 3 der I/Q-zu-ZF-Transformation



LabVIEW FPGA-Quellcode des Splitters



Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

München, den 12. Mai 2015

Stefan Sailer